Андрей Орлов

Записки автоматизатора

Профессиональная исповедь

Записки автоматизатора
Андрей Орлов

SHOUT
SHARE
IT

Annotation

Личные советы опытного разработчика и IT-менеджера «специалистам, приступающим к установке компьютерной системы учета чего бы то ни было», а также заказчикам такой системы.

- Предисловие
- Благодарности и представление автора комментариев
- Краткая характеристика лирического героя
- Вас зовут на работу
- Перед тем как согласиться
- Вы согласились
- Выбор системы
- Подбор персонала
- Стиль руководства
- Формулирование задач
- Траблы-грабли-бумс!
- Шизофреническая глава про абсолютно нормальных людей
- Финишная кривая
- Заключение
- Приложения
  - Приложение 1. Разбор задачи для собеседования с программистами
  - Приложение 2. Из записных книжек периода внедрения
  - Приложение 3. Страшный сон в приказах
- Приложение 4. Когда машины были большими
- Об авторе
- notes
  - 1
  - 2
  - 3
  - 4
  - 5

Предисловие

С момента появления первого варианта этого текста прошло почти десять лет, но он все так же не претендует ни на полноту охвата рассматриваемых вопросов, ни даже на какую-либо объективность. С появлением моей страницы в Интернете он перекочевал туда с моего личного харда под названием «Записки автоматизатора», потом расселился по всей Сети (причем отнюдь не всегда с моего ведома), потом появился частично в журнале «Технологии лизинга и инвестиций» (№ 1 (12), 2004), потом в почти полном на то время, но очищенном от неприличных слов виде был опубликован в журнале «Бухгалтер и компьютер» (№№ 4 (67), 6 (69), 7 (70), 2005). Как ни странно, текст все еще продолжают читать, хотя системы, которые я выбирал, когда начинал его писать, уже выкидывают.

Все это время текст служил мне для абсорбции яда, выделяемого в качестве побочного продукта при внедрении информационных систем, дабы он не попал на окружающих, и помогал сформулировать что-то для себя. Но если то, что написано ниже, полезно другим участникам увлекательного процесса автоматизации предприятий – от генерального директора до оператора по подготовке данных – и иногда позволяет им не наступить на часть ожидающих их грабель, то это очень приятно.

Эта книга не учебник. В ней нет ни определений, ни попыток связно изложить какие-то дисциплины, составляющие бэкграунд специалиста по информационным технологиям, или их разделы. Скорее, это попытка передать мой личный опыт людям, которые уже находятся в теме или должны находиться в ней в соответствии со своим служебным положением, уже обладают собственным опытом в нелегком деле создания и внедрения информационных систем или их составляющих, но все еще хотят узнать больше.

Я старался избегать специальной терминологии во всех случаях, когда она служит исключительно для затемнения смысла и демонстрации превосходства айтишников над простыми смертными, но оставлял ее в тех случаях, когда без нее, на мой взгляд, не обойтись. Поэтому вы не встретите здесь пермутации экстентов, манданты и деплои, но репликации и коллизии все-таки попадутся, уж простите.

Очень прошу не воспринимать эту книгу как скрижали завета. Если читатель обдумал любое изложенное здесь утверждение и у него сформировалось мнение, противоположное мнению автора,

то автор считает свою задачу выполненной.

Еще эта книга не учит выдувать мыльные пузыри. Есть такая ИТ-специализация: изготовлять муляжи проектной документации и имитации внедрений в целях получения откатов, кредитов, повышения привлекательности и капитализации компании и т. п. Мне этим тоже иногда приходилось заниматься, но передавать свой опыт в таких делах я не собираюсь.

Я буду исходить из несколько наивного предположения, что информационная система внедряется, чтобы руководство компании получало точную и корректную информацию, а не пудрило мозги кому бы то ни было. Ну, разве что немножко.

Благодарности и представление автора комментариев

Рукописью (а точнее, вордописью) этой книги я изнасиловал достаточно много знакомых и не очень знакомых специалистов и не только специалистов. В результате был получен большой объем замечаний, предложений и комментариев. Хотя правка родного текста по замечаниям заставила меня вспомнить те ощущения, которые я испытывал, приводя своего маленького сына к зубному врачу, некоторые изменения я все-таки внес. За эту операцию я приношу свои глубочайшие благодарности Галине Демич, Марине Денисовой, Геннадию Чернецкому, Юрию Медведю, Леониду Ленесяну, Артему Федяю, Евгению Пясецкому, Андрею Журавлеву, Андрею Кравченкову, Владимиру Самодурову и всем остальным, кого забыл вспомнить по причине хронического склероза. Консультации Вадима Мирного помогают мне всю жизнь. Эта книга тоже без них не обошлась.

В самой первой версии «Записок автоматизатора», выложенной на моем интернет-сайте, я в одном из мест поступил как истинный айтишник, влезающий в бухгалтерию, а именно: перепутал дебет с кредитом. Из тысяч прочитавших текст только трое обнаружили ошибку и не поленились о ней мне написать. Одним из написавших был Дмитрий Комаровский. Когда текст «Записок» после многочисленных трансформаций начал превращаться в книгу, я списался с Дмитрием, как и со многими другими, кто проявил интерес к этому тексту, с просьбой оценить результаты моего труда.

Ответ пришел в виде вордовских примечаний, которые меня озадачили. Все было написано по делу, явно профессионалом, но… это никак не встраивалось в мой текст. Я этого точно не мог написать. Автор примечаний внимательно рассматривал те же объекты, что и я, но под другим ракурсом. Поэтому картинки, которые он рисовал, успешно дополняли мои, но совмещать наши картинки в одном изображении не получалось.

Мне кажется, что я пришел к единственно правильному решению: оставить все комментарии там, где они были поставлены, и в том виде, как их вставили. И результатом лично я доволен.

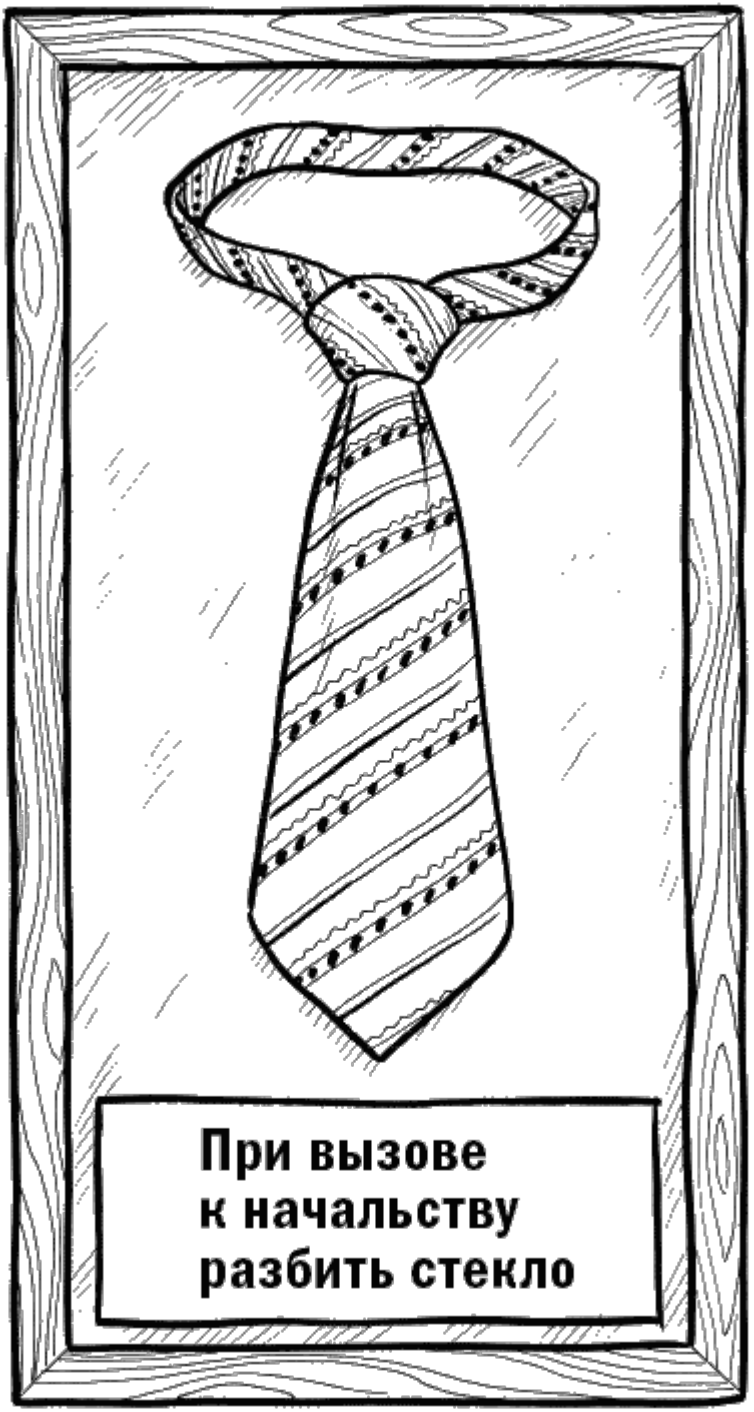
Как ни странно, человека, столь ответственно отнесшегося к моему тексту, я никогда в жизни не видел и не знал о нем ничего, кроме того, что он лучше представляет бухгалтерские проводки по счету номер 99, пока не получил от него еще один

комментарий, который привожу ниже. Остальные комментарии Дмитрия Комаровского дальше во всей книге выделены таким же способом.

> Я родился 31 января 1977 года в Ленинграде (там же и живу, только он теперь Санкт-Петербург). Учился в Инженерно-экономической академии по специальности «экономист». Рабочую карьеру начал в 90-е годы, сначала как бухгалтер, затем как консультант-автоматизатор. В качестве автоматизатора застал век заката больших машин на больших заводах, наблюдал расцвет 1С и великое нашествие западных ERP-систем (после чего долгое время считал слово «локализация» ругательным). Работал как менеджер проектов со стороны компаний – «системных интеграторов» на проектах внедрения больших и малых ERP-систем (поневоле участвуя в локализации оных) в больших и малых компаниях (где застал такие чудеса и удивительности, какие разработчикам этих систем и не снились).
>
> В настоящее время перестал работать на поставщиков программного обеспечения и работаю в крупной консалтинговой компании, специализируясь на консалтинге в области внедрения ERP. – *Д. К.*

Кроме тех, кто научил меня ремеслу, и тех, кто принимал деятельное участие в создании и продвижении этой книги, я, наверное, должен поблагодарить всех дураков, неучей и мерзавцев, которых встречал на своем рабочем пути, поскольку именно они дали для этой книги максимальное количество материала.

Но я не буду делать это персонально.


При вызове
к начальству
разбить стекло

Краткая характеристика лирического героя

– База, база! Я Хабибулин. Кто я?

Из анекдота

Я написал свою первую программу в 1973 году. Правда, понимание того, как это делать лучше, пришло ко мне гораздо позже.

Советские времена были благословенны для автоматизаторов. По причине одинаковости зарплат работали мы там, где люди были приятнее, а работа интереснее. А что и как автоматизировать, мы выбирали сами. Начальству это было не очень важно. Иметь систему в организации было модно, дисплей в кабинете престижно. А цифры… Кого они волновали? Но для того, чтобы ваше начальство могло успешно втирать очки своему, вы должны были предусмотреть ввод данных для итоговых отчетов вручную, независимо от информации, которая уже имелась в системе.

Теперь системы заказывают люди, которые тратят на них свои деньги (или почти свои, если речь идет о государственном или муниципальном заказе). И за свои деньги хотят что-то получить. Всегда ли они представляют, что именно, это отдельный вопрос.

Возраст мой с момента погружения в ИТ еще даже не утроился, но частоты используемых процессоров увеличились за то же время в миллионы раз, а объемы памяти – в миллиарды. Да, я начинал на ЭВМ, у которой оперативной памяти было 1024 ячейки, а из внешних накопителей информации была только перфолента. И я еще успел вживую услышать рассказ Роальда Аркадьевича Мирного про то, как в свое время к ним на работу привезли машину, в которой оперативной памяти было целых двести ячеек, и они ломали голову, как можно использовать такой немыслимый объем.

Мне уже сложно иногда вспомнить, какими аббревиатурами обозвали очередную новомодную дырку в компьютере и то, что в нее засовывают. Да, за это время поменялось многое. Главное, появились не только прибамбасы и навороты, но и новые методы разработки и программирования. И теперь плохо написанный алгоритм будет работать не сорок миллиардов лет, как в моей молодости, а всего пару тысяч. Но кирпичи, подброшенные над

своею головой, все так же падают на голову, хотя материал, из которого они сделаны, стал другим.

Первый проект, которым я руководил «по-взрослому», а не в рамках научно-технического творчества молодежи, громко назывался «Подсистема АСУЖТ “Инженерно-технические кадры железнодорожного транспорта”». В команде было семь человек из двух организаций. Разработка с опытной эксплуатацией заняла три месяца, исходные данные нам присылали на колодах перфокарт из управлений 33 железных дорог Советского Союза. Внедрение сопровождалось личным рекордом непрерывной работы в 26 часов. Было это в 1979—1980 годах. Рекорд 2003 года – 36 часов подряд. Но что-то мне эти эксперименты в последнее время нравиться перестали: уж больно потом крыша едет.

Достаточно давно я не задерживаюсь на одном месте работы больше пяти лет. Хотя ни разу в жизни меня с работы никто не увольнял. Просто в какой-то момент понимаешь, что, оставаясь на прежнем месте дальше, заплесневеешь уже на всю оставшуюся жизнь.

Спрос на меня все еще есть, благо мозги еще не совсем засохли, а профессиональную репутацию, формировавшуюся лет тридцать, я еще не испортил. Эта профессиональная репутация сводится к четырем фразам:

– я умею внедрять информационные системы;

– я умею учиться;

– я никогда не обманывал, не подставлял и не продавал работодателя;

– я не запоминаю никаких чисел, кроме размера своей зарплаты.

Нанимают обычно организации, созревшие для перемен и сознающие, что эти перемены очень сложно провести людям, находящимся «снаружи». Предпринимателям сложно пока доверить консультантам и интеграторам все свои секреты. А поверить, что консультанты будут отстаивать интересы заказчика, а не свои собственные, еще сложнее.

Кроме того, кто бы ни реализовывал проект автоматизации, именно внутри организации должен находиться хотя бы один человек, который может понять и оценить, что же делают все внешние консультанты и внедренцы. А к моменту внедрения в организации должно существовать подразделение, которое будет сопровождать внедренную систему, а его обычно тоже надо создавать, что снаружи делать уже совсем несподручно.

Стандартный цикл жизни в новой компании обычно начинается без подчиненных или с одним-двумя таковыми, а заканчивается, когда я оставляю работоспособное подразделение

в десять-двадцать сотрудников.

Слух о том, что я могу внедрить что угодно где угодно, не соответствует действительности, хотя и лестен: я просто отказываюсь от предложений, если получение результата заведомо невозможно.

Известен случай, когда менеджер по продажам тиражируемой информационной системы предлагал меня вместе с этой системой. За систему деньги получала фирма-разработчик, за меня он требовал немалое комиссионное вознаграждение себе. Менеджеру пришлось через некоторое время уволиться: систему он сторговал в три места, а я почему-то пошел на работу только в одно из них. В остальных система внедрена не была. Мне рассказали о стоимости моего трансфера через три года после того, как он произошел.

Вас зовут на работу

Итак, вы – высококлассный профессионал-постановщик и получили предложение работы. Приглядитесь к фирме, сделавшей предложение, и попытайтесь понять, на какой стадии развития она находится (правда, если вы не догадались сделать это сами, какой же вы, к черту, «высококлассный профессионал-постановщик»?).

Стадии жизненного цикла компании

Стадия первоначального накопления капитала

Мне все равно, как это было. Может быть, даже не на большой дороге. Главное, что у людей слишком часто формируется впечатление, что деньги на них сыплются из рога изобилия.

Помимо остальных средств роскоши (в зависимости от вкуса и культурного уровня это могли быть золото-брильянты, «Феррари» или шестисотый «Мерседес», личный самолет и т. п.) в это время закупаются крутые компьютеры, сетевые операционные системы и СУБД. В качестве одного из таких средств, так греющего душу, на работу берутся высокооплачиваемые специалисты по информационным технологиям, желательно такие, которыми можно похвастаться перед знакомыми. Выбор может пасть на человека, известного в России своими программными продуктами или статьями. В крайнем случае годится преподаватель вуза, который тебя учил, да еще и двойки ставил.

Осмысленность происходящего мало кого волнует: когда деньги сыплются, главным инструментом является не калькулятор, а лопата.

Никто не может объяснить, зачем менеджерам и операторам мультимедийные компьютеры с навороченными мониторами, звуковыми картами и активными колонками. Наверное, чтобы компьютер играл туш при кредитовом сальдо бухгалтерского счета «Прибыли и убытки» и похоронный марш при дебетовом.

Когда денег много, редко кто удосуживается посчитать их точно. И если все-таки количество денег, которое должно попасть в карман, тем или иным способом подсчитывается, то вот желания оценить, сколько денег из общей кучи тебе приносит какое-либо подразделение, не бывает почти никогда.

Поступать на работу в компанию в это время экономически

выгодно. Только может быть скучно, так как вы обнаружите, что руководство забыло о вашем существовании, и немножко нервно, так как вы будете представлять, что же произойдет, когда оно наконец о вас вспомнит.

Когда запахло жареным

Все изменяется тогда, когда приходит понимание, что прибыль уменьшается или вот-вот должна уменьшиться. Это следующий момент, когда обычно обращаются к информационным технологиям.

Заветная надежда, связанная с автоматизацией в это время, была высказана в начале 80-х годов одним из заместителей министра путей сообщения во время выступления перед ИТР некоего министерского КБ: «Вы же все здесь очень талантливые люди. Ну, сделайте такое электронное устройство, чтобы пропускная способность наших железных дорог увеличилась…» Талантливые люди очень вежливо посоветовали прокладывать дополнительные пути, чтобы было где ездить дополнительным поездам. С тех пор я успел поработать во многих местах, но смысл просьб руководства зачастую сводился к тому же. Наверное, все дело в том, что эти люди атеисты. Иначе бы они молили Бога. Хотя, по имеющейся у меня информации, Господь тоже редко помогает увеличивать прибыльность бизнеса без капитальных вложений.

> Возможно, одна из причин этого в том, что мы называем все, что устанавливается на компьютер, словом «система». Наверное, правильнее было бы называть это просто компьютерной программой, так как «системой» оно станет только тогда, когда превратится в часть внутренних процессов компании.
>
> Но зато «система» куда как лучше продается. Даже в стиральном порошке теперь используются не химические соединения, а «система супер-пупер-очистки». – *Д. К.*

Попытки объяснить, что постановка учета и получение дополнительной информации **никогда** не дают прямого эффекта, что не получится сэкономить деньги на фонде заработной платы, что за дополнительную информацию деньги приходится платить всегда, а чтобы выжать из нее свою прибыль, нужно еще очень сильно поработать собственной головой, и никакой отдел автоматизации тут ничего сделать не сможет, если не начнет подменять топ-менеджмент фирмы в его непосредственной работе, могут восприниматься спокойно, даже благожелательно (вот какого умного наняли, можно ему еще зарплату добавить),

но когда внедрение прошло, вас все равно удивленно спросят, почему же и теперь нельзя никого уволить.

В этот момент денег вам пообещают тоже много, но нужно понимать, что период их выплаты для вас сильно сокращается. Однако если вакансия, которую вы ожидаете, должна, по вашим расчетам, открыться через полгода, это способ пересидеть необходимое время, хотя это может отрицательно сказаться на вашей профессиональной репутации.

Когда крах близок

На этом этапе начинаются судорожные попытки разделить бюджеты подразделений или отдельных компаний холдинга, чтобы решить, кому умирать раньше. Лишь в этот момент обычно доходит, что похлопывание по карману является не лучшим способом оценки прибыли. Тут же выясняется, что ни на разработку, ни на закупку, ни на внедрение соответствующих программных средств именно теперь нет денег. Да и времени на все это тоже нет.

После нескольких конвульсий наконец удается сформулировать великое правило: **стоимость системы расчета прибыли не должна превосходить эту прибыль**.

И все информационные проекты замораживаются или сокращаются.

Боже вас упаси соваться на фирму в этот момент! В лучшем случае вы не получите ничего, кроме обвинений, что фирму развалили именно вы.

Понимание цели

Если состояние компании вас удовлетворило, следует понять реальные ожидания нанимателя при вашем приеме на работу. Они могут быть различны:

1) наладить в компании правильный учет (имеется в виду, что вы знаете, что такое правильный учет и готовы подсказать это нанимателю);

2) наладить в компании ровно такой учет, какой хочет наниматель (это бывает очень забавно);

3) снабдить офис всякими электронными игрушками, которыми наниматель будет хвастаться перед своими знакомыми;

4) повысить инвестиционную привлекательность компании за счет внедрения крутой и модной информационной системы.

Учтите, что сам наниматель может считать вашей целью п. 1, но заставлять вас реализовывать п. 3, поскольку искренне верит,

что учет состоит именно из игрушек.

Но в общем случае, поскольку это вы поступаете на работу, а ваш наниматель уже имеет собственный бизнес, иногда можно предположить, что наниматель разбирается в бизнес-стратегиях лучше вас. А бизнес-стратегии могут быть весьма разными и предполагать, например, экономию средств на ИТ, которая достигается организацией учета не с точностью до копейки, а с точностью до тысяч долларов. Я встречал компании, работавшие и по таким принципам, они были вполне успешными и не разорялись в отличие от компаний, где хозяин узнавал о каждой продаже в каждом из двухсот магазинов по звуковому сигналу, который издавал его компьютер.

Еще нужно постараться понять, нет ли среди руководства компании «вселенского ИТ-мечтателя«, жаждущего запихнуть в компьютер весь реальный мир и на меньшее не согласного. Переходить можно, только если «вселенским мечтателем «является хозяин фирмы и он достаточно богат. Внедрения, конечно, не будет, но вы будете творить нетленку и получать зарплату, пока у него не кончатся деньги.

Работа в империях

При должном уровне оплаты работа в государственных и окологосударственных организациях и очень крупных корпорациях на первый взгляд кажется удачным выбором. Кажущаяся стабильность этих монстров подкрепляется официальностью всей зарплаты, а не меньшей ее части, наличием медицинских страховок и возможностью реализовывать глобальные проекты.

Однако всему этому есть и тяжелый противовес: у каждого представителя гигантской руководящей надстройки таких империй есть свои задачи и приоритеты, отнюдь не всегда совпадающие с задачами и приоритетами организации. Не секрет, что чиновника на Руси издревле гораздо больше волновали вопросы продвижения по служебной лестнице, прикрытия собственной задницы и личного обогащения (в случае ИТ в основном за счет откатов), чем государственные и корпоративные интересы. Поэтому вовсе не факт, что начальник, который хочет взять вас на работу, собирается внедрить информационную систему. На самом деле он может хотеть:

– создать видимость активности на ИТ-фронте, но сохранить непрозрачность основных показателей деятельности, поскольку его смогут выгнать или посадить, как только основные

показатели можно будет проанализировать;

– получить откат за приобретение дорогих программных средств и железа, потом признать приобретение всего этого вашей ошибкой, выгнать вас, нанять следующего специалиста и повторять эту операцию, пока самого не выгонят;

– с вашей помощью изгнать неугодного сотрудника как неэффективного, после чего потерять к вам интерес;

– просто занять вами вакансию в штатном расписании, поскольку это повышает его статус.

Внедрять в таких местах крупные информационные системы всегда чрезвычайно тяжело, поскольку по большому счету они никому не нужны и не интересны. Вместо одной понятной цели, к которой вы должны стремиться в своей работе: повысить обороты или прибыль компании, в крайнем случае развлечь ее хозяина, — вы получаете установки, высказанные намеками, да еще и разные от разных начальников. А начальников в таких местах много. Периодически вам приходится уворачиваться от чугунной бабы, запущенной царственной рукой в голову неугодного, но попутно сметающей и другие головы, не успевшие пригнуться.

> *Одна сцена работы в компаниях такого рода до сих пор стоит у меня перед глазами. Совещание о ходе внедрения информационной системы проводит заместитель Генерального (именно так полагалось писать: заместителя – с маленькой, а генерального – всегда с большой). Я сообщаю, что мы не можем продвинуться в формировании справочника контрагентов, поскольку одно из подразделений не указывало ИНН у организаций, с которыми работало. А поскольку организации эти имели очень распространенные в России названия (например, совхоз «Рассвет»), то сейчас невозможно понять, с одним ли «Рассветом» мы имели дело двадцать раз или по одному разу с двадцатью разными «Рассветами». Впрочем, задача решается, поскольку в бухгалтерии есть все необходимые документы, из которых можно получить информацию. Но кто-то этим должен заняться.*
>
> *Я сижу за одним торцом большого стола для заседаний, замгендир – за другим. По правую его руку располагается начальник того самого подразделения, а по левую – представитель бухгалтерии. Но когда я прекращаю говорить, ожидая его реакции, он, стараясь не посмотреть ни направо, ни налево, поднимает глаза над моей головой и произносит: «Это нужно сделать». То есть*

я теперь не могу сказать, что он не выдавал соответствующего задания. Но и выполнять его никто не будет, поскольку оно ни к кому лично не адресовано.

Я все понял и не стал спрашивать, кому это нужно сделать. Я начал подыскивать новое место работы.

Правда, один плюс при внедрении просматривается: к документальному оформлению всяческих технологий и регламентов документооборота в таких местах относятся с большим пониманием и пиететом.

Что еще лучше понять сразу

Руководители, которые не хотят иметь информации, встречаются очень редко и, что для нас самое главное, не заказывают информационных систем. Другое дело, что даже тот руководитель, который заинтересован в получении информации, обычно совсем не расположен к тому, чтобы этой информацией с кем-либо делиться.

Нежелание делиться полной и объективной информацией с конкурентами, налоговыми органами и даже собственными сотрудниками понятно и всегда учитывается при проектировании систем.

Однако нежелание исполнительного директора делиться информацией с генеральным, директора по снабжению – с директором по сбыту и т. д. может похоронить внедрение любой информационной системы.

Традиционная паническая боязнь хозяев показать кому бы то ни было свои доходы и прибыли приводит к несколько другому эффекту. Информация в системе собирается, но не анализируется: кто имеет – не может, кто может – не имеет.

И пожалуйста, никогда не путайте своего работодателя со своим другом, даже если это одно и то же лицо: когда первый сократит вашу зарплату до такого уровня, что вам нечего будет жрать, второй обязательно предложит денег взаймы

Перед тем как согласиться

Главный автоматизатор (начальник департамента/ управления/отдела автоматизации/ информационных технологий, ИТ-менеджер, сейчас вошло в моду новое слово – CIO), как бы он ни назывался, – не грузчик. Поэтому прием его на работу не должен происходить после одного интервью. Кстати, если ваши работодатели не понимают смысла первого утверждения, лучше поискать себе другую компанию.

> Что менее очевидно — основной деятельностью такого человека является отнюдь не закупка компьютерного железа. – *Д. К.*

Ваша профессия высокооплачиваемая и, что еще хуже при поиске работы, штучная. Поэтому лучше потерпеть еще немного, чем снова оказаться в подвешенном состоянии через месяц после смены работы. Не пытайтесь получить место при активном нежелании хотя бы одного из топ-менеджеров. Поймите, что такой человек всегда сможет доказать конструктивно, что вы не справляетесь со своей работой.

Если вы почувствовали заинтересованность в себе, не бойтесь перечислить хотя бы устно свои требования к организации работы, без которых вы не сможете выполнить поставленные перед вами задачи. Сто раз повторите, что вы не волшебник, а специалист по автоматизации. Покажите руководству прилагаемую таблицу. Но не забывайте, что все это ни от чего вас не защитит.

Чего не может понять большой начальник

Очевидные утверждения, с которыми все обычно согласны	Очевидные следствия, которые обычно недоступны пониманию
1. Если что-то сосчитать, больше его от этого не станет. Но если не сосчитать, то может стать сильно меньше, особенно в России.	Учет сам по себе не может увеличить прибыли. Он позволяет только: — затруднить воровство; — позволить принять правильные бизнес-решения. К сожалению, последнее работает только при достаточной оперативности и точности информации и (что самое главное) при наличии мозгов у принимающего решение.
2. Сосчитать что-то всегда стоит дороже, чем не считать.	Автоматизация учета никогда не приводит к уменьшению затрат на учет.
3. Необходимым условием успешного завершения автоматизации является сильное желание первого руководителя.	Необходимым условием успешного завершения автоматизации является наличие первого руководителя в каждом подразделении.
4. Результатом автоматизации бардака всегда становится автоматизированный бардак.	Если вы не в состоянии сами руководить своей фирмой, то автоматизация учета вам не поможет.

Вообще начальству надо говорить правду, всю правду и ничего, кроме правды. Обычно это не приводит ни к каким печальным последствиям. Потому что в принципе ни к каким последствиям не приводит. Все, что вы сообщите начальству, оно либо пропустит мимо ушей, либо не примет на свой счет, либо согласится, но поймет не так, как вы хотели. Поэтому не стесняйтесь выражать свои мысли в письменной форме, даже если это не принято в компании. Впоследствии можно будет достать из стола (повторно распечатать) ту же бумагу и… испытать хотя бы моральное удовлетворение.

Текст, приведенный ниже, тоже неплохо показать. Как я понял на собственном опыте, изложенное в нем не всегда принимается обеими сторонами, заключающими трудовое соглашение, по умолчанию. В какой-то момент именно этот текст горячо обсуждали на различных форумах, связанных как с ИТ, так и с подбором кадров. И обсуждения эти были иногда весьма забавны. Из лучшей реплики я узнал, что меня не существует в природе, а кодекс придумали злобные работодатели, чтобы таким способом захомутать своих работников.

Моральный кодекс наемного работника

1. Я осознанно выбрал роль наемного работника, во всяком случае на этом этапе своей жизни.

2. Я сформулировал этот кодекс прежде всего для себя, так как он облегчает мне жизнь и упрощает мои взаимоотношения с работодателями.

3. Я применяю этот кодекс независимо от того, верят ли мне, что я его применяю.

4. Я не продаюсь в рабство, а поступаю на работу в фирму в соответствии с достигнутыми между мной и фирмой договоренностями, налагающими определенные обязательства как на меня, так и на нее.

5. Фирма, в которой я работаю, – мне дом родной. Пока я в ней работаю.

6. Я во всех случаях сохраняю к ней лояльность, отстаиваю ее интересы, не ворую, не беру взяток, откатов, подарков и не веду деятельности, которая может повлечь за собой ущемление ее интересов.

7. Вместе с тем даже в интересах фирмы я не пойду на нарушение моральных и этических норм, мною признаваемых.

8. Я работаю на фирму, а не на конкретное лицо в ней, независимо от ранга этого лица.

9. Я отделяю своих друзей и родственников от своих руководителей и подчиненных, даже если это одни и те же люди.

10. Я не работаю на другие фирмы, если это было оговорено при приеме на работу, но всегда сохраняю за собой право на свободную творческую деятельность, если обратное не было оговорено особо.

11. Коммерческие тайны, доверенные мне фирмой, я не разглашаю и после своего увольнения.

12. Я стараюсь повысить эффективность работы фирмы во всех случаях, когда могу это сделать, независимо от того, что написано в моей должностной инструкции. То, что поручено мне, я стараюсь сделать хорошо сам; в остальных случаях я даю рекомендации по улучшению работы тем руководителям, которые хотят меня выслушать. Если мои предложения не принимаются, я стараюсь оптимизировать работу в тех случаях и на тех участках, на которых могу.

13. Я всегда информирую руководство о реальном состоянии дел и реальных прогнозах, даже если

руководство этого и не хочет.

14. Я соблюдаю дисциплину и субординацию, принятую в фирме.

15. Я никогда не пытаюсь доказывать свою правоту деструктивными способами типа «сделаю все в точности так, как ты сказал, а когда все рухнет, ты поймешь, кто был прав».

16. Если я не главный управляющий фирмы, я отдаю себе отчет в том, что, хотя успехи и неудачи фирмы до некоторой степени и зависят от моих усилий, но не в полной мере ими определяются.

17. Тезис «Проблемы фирмы – мои проблемы» я не принимаю абсолютно, но только вместе с тезисом «Мои проблемы – проблемы фирмы».

18. Я всегда стараюсь выполнить обязательства, данные мной при приеме на работу или в процессе работы, если фирма выполняет обязательства, данные мне.

19. Я по возможности не продолжаю трудовые отношения с фирмой, если, на мой взгляд, она реально не нуждается в моих услугах, даже если меня никто не гонит.

20. Я отдаю себе отчет и не скрываю от руководства фирмы, что эффективно могу работать в одном направлении не более лет пяти, после чего буду вынужден поменять сферу деятельности или фирму.

21. Я работаю в фирме на определенных условиях (оплата, режим работы и т. п.), оговоренных при приеме на работу или общепринятых, и в определенных условиях (помещение, климат, возможности питания). Если эти условия перестают выполняться или меня устраивать (по уровню ли зарплаты, количеству выделяемого мне на работе кислорода, состоянию физического или морального климата и даже собственного состояния), я могу покинуть фирму.

22. Я заранее предупреждаю руководство о своем желании покинуть фирму и, если руководство того хочет, обсуждаю с ним условия, при которых я останусь.

23. Если же руководство фирмы без предварительного уведомления снижает мою зарплату или резко ухудшает условия труда, я считаю себя вправе покинуть фирму также без предварительного уведомления.

24. Я никогда не шантажирую руководство угрозой

своего увольнения.

25. Если я объявил о своем увольнении, то я увольняюсь, не обсуждая предложений, которые поступили после этого объявления.

26. Перед увольнением я стараюсь сдать дела своему преемнику, если могу его обнаружить, и делаю все возможное для того, чтобы мой уход не повлиял на состояние дел на фирме отрицательно.

27. Однако, если при расставании часть моей работы останется неоплаченной, я считаю себя вправе не допустить использования результатов этой работы. Но без поджогов и диверсий.

28. Я никогда не делаю специальных гадостей фирме, в которой работал, после своего увольнения:

– не занимаюсь ее очернением;

– не переманиваю сотрудников только с целью их ухода из фирмы;

– не разглашаю ее коммерческих тайн;

– не нарушаю ее имущественных, авторских и других прав;

– не доношу и не навожу на нее;

– не закладываю в компьютеры вирусы, не форматирую диски и не занимаюсь уничтожением или сокрытием информации любыми другими способами;

– не использую новое место работы для нанесения ущерба предыдущему.

29. Вместе с тем я не отношу к деятельности, перечисленной в предыдущем пункте, и потому оставляю за собой право:

– высказывать свое мнение о состоянии дел на оставленной фирме и характеризовать ее персонал;

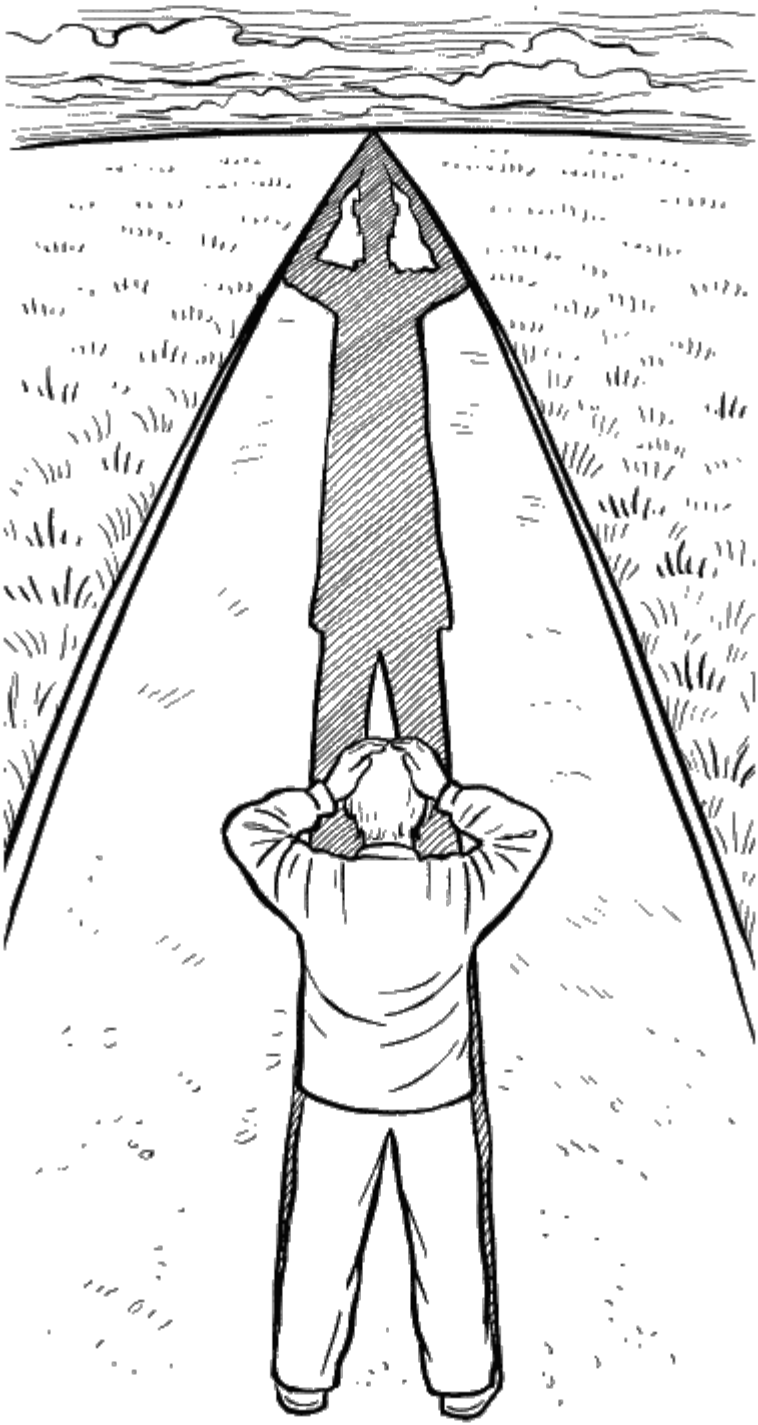
– приглашать на работу сотрудников оставленной мною фирмы на новое место работы с предложением лучших условий работы или оплаты труда, если эти сотрудники мне действительно нужны: после увольнения я становлюсь для оставленной мной фирмы равноправным конкурентом на рынке рабочей силы.

30. Я стараюсь отказаться на новом месте работы от выполнения функций, которые вступают в противоречие с интересами оставленной фирмы, но, если это невозможно, действую в интересах своей новой фирмы. Потому что фирма, в которой я работаю, – мне дом родной. Пока я в ней работаю.

Разумеется, это мой кодекс. Перед тем как показывать его потенциальным работодателям, из него нужно вычеркнуть положения, которые вы не собираетесь выполнять.

И последний совет этой главы (правило Галины Демич): **договорившись о размере зарплаты, не забудьте отдельно договориться, что вам ее будут платить.**

Вы согласились



Осмотритесь

В первую же неделю работы необходимо составить и представить руководству документ, определяющий ваши функции (функции руководимого вами подразделения). Что-то типа «Основные функции и распорядок работы отдела ФИТ (фальсификационных и информационных технологий)».

Если вы этого не сделаете, то еще через неделю набранные вами системные инженеры начнут чинить холодильники в магазинах фирмы и магнитолы в хозяйских автомобилях. Сложнее придумать, как использовать не по назначению программистов, которые редко умеют делать что-то руками, но фантазия начальства в этом направлении не знает границ.

Документ должен содержать исчерпывающий перечень обязанностей сотрудников отдела и принципы построения графиков работы. На мой взгляд, оптимальной является фраза «Режим работы сотрудников отдела должен обеспечивать полное выполнение возлагаемых на отдел функций. Составление индивидуальных графиков работы возлагается на руководителя отдела». В этом же документе неплохо оговорить, какие категории сотрудников имеют ненормированный рабочий день, а каким оплачиваются переработки и как именно они оплачиваются. Это, может быть, снимет часть неизбежных дальнейших конфликтов.

По возможности документ должен четко оговаривать взаимодействие отдела с другими подразделениями и определять, кто имеет право ставить вам задачи и кто обязан их с вами обсуждать, а также что делать, если вам не дают необходимой информации.

По моим расчетам, какие-нибудь из правил налогообложения изменятся не позднее чем через месяц после начала опытной эксплуатации системы, а через полгода Госкомстат поменяет все формы первичных документов и отчетности. Если вы не оговорите, кто обязан заранее информировать вас обо всех изменениях в законодательстве, то сможете поучаствовать в оплате штрафов, наложенных на вашу фирму налоговой службой или торговой инспекцией.

Теперь вы можете приступить к «обследованию фирмы для выяснения информационных потребностей и постановки задачи».

Прежде всего попробуйте составить для себя небольшой справочник «Кто есть кто в компании» и определить, кто принимает решения. Если вам не удалось сделать это в течение первой недели или вы обнаружили, что источником решений является мизинец левой ноги хозяина, то прогноз вашей

дальнейшей работы печален.

Классификация хозяев и управляющих

В математике есть функции, не являющиеся ни четными, ни нечетными. И таких большинство. Честное слово, я встречал достаточно много хозяев и руководителей, которые под мою классификацию не подходят, я даже работал с такими, но что о них можно написать? Ну стараются они затраты минимизировать обычно, прибыли увеличить, специальные вопросы со специалистами обсуждают, внимают голосу разума и даже чужой интуиции (но это если человека уже узнали). Так что дальше описаны только типажи, про которых есть что написать, но с которыми очень сложно добиться каких-нибудь результатов в работе.

Изменение статуса человека (состояния, дохода, должности, положения в обществе и т. п.) достаточно часто приводит к психическим нарушениям, при которых внешний мир перестает восприниматься адекватно. Происходит это на всех уровнях иерархии (студент, давший взятку за оценку на экзамене, вдруг уверяется, что он прекрасно знает предмет, поскольку у него «пять»; программист, которого сделали начальником отдела разработки, но потом в ужасе от его некомпетентности сняли, начинает считать себя знатоком в создании информационных систем предприятия), но похоже, что с хозяевами чаще. К сожалению, у работодателей не принято требовать справку от психиатра и нарколога. Впрочем, в нашей стране наличие любой справки говорит лишь о том, что у предъявителя были на нее деньги.

Небожитель. Я не про того хозяина, который сам не занимается бизнесом, в котором вы задействованы. Интересы такого обычно представляет управляющий менеджер, а прямые воздействия с неба, как правило, не производятся. Я про того хозяина, который регулярно посещает офис. Его можно встретить в коридоре, столкнуться в дверях туалета, поздороваться, даже обсудить погоду или общих знакомых. Но с ним почему-то оказывается совершенно невозможно поговорить о работе. Хотя, как говорят, основные бизнес-решения принимает именно он. Но все такие решения доводятся до офисного планктона через одного или нескольких жрецов-толкователей. При этом сами толкователи в силу своего статуса не уполномочены принимать решения. В результате все решения небожителя не обсуждаются и не подлежат изменению: если ты пытаешься объяснить жрецу, почему очередная булла не слишком хороша, то получаешь ответ: «Я Ему говорил то же самое, но Он стоит на своем».

Гений. Обидно, что обычно это люди на самом деле совсем не глупые, но мучимые жаждой быть непревзойденными во всех сферах деятельности. И на уровне компании свою жажду удовлетворяющие очень простым способом: для руководства всеми направлениями набираются сотрудники, разбирающиеся в этих направлениях хуже набирающего. Эффект достигается достаточно быстро.

Развлеканец. Ключевая фраза, по которой можно отличить этот тип хозяина: «Из-за этого я теряю пять тысяч долларов каждые пятнадцать минут». Время и сумма могут варьироваться – неизменно одно: если вы начнете приставать к такому с просьбой привести расчеты, на которых строится это утверждение (чего я делать очень не рекомендую), то расчетов не дождетесь, но через некоторое время приведете развлеканца в бешенство с тяжелыми последствиями для себя. Традиционное поведение бизнесмена у представителей этого типа претерпело одно маленькое изменение: вместо установок «Заработать кучу денег и развлечься на них» или хотя бы «Развлечься зарабатыванием кучи денег» работает установка «Развлечься своим бизнесом». Им главное, чтобы было не скучно.

Развлеканцы обычно люди чрезвычайно неглупые и энергичные, а ведение успешного бизнеса включает слишком много рутины. Чтобы развлечься, можно заставить своих сотрудников работать по ночам, как это сделал отец всех народов, можно отменить отпуска, можно менять правила торговли каждый день, можно гнобить одно направление бизнеса и раздувать другое. Аргументация во всех случаях принятия решений приводится интуитивная: «Мне кажется, что…», «Я абсолютно уверен…» Подтверждение своей интуиции всегда можно найти, если после фразы «Я точно знаю, что у этого бизнеса нет перспектив» изъять из указанного бизнеса оборотные средства. Характерной особенностью является также абсолютная уверенность при реализации наиболее бредовых своих идей (чужими руками, разумеется): «Пилите, Шура, пилите. Я точно знаю: они золотые».

Один из известных мне развлеканцев доводил своего финансового контролера, в задачи которого входил расчет чистой прибыли за период на основе имевшихся данных учета денег и материальных ценностей:

– И сколько мы заработали?

– Двадцать.

– А мне кажется, что все-таки пятьдесят, – и шел в кассу забрать деньги себе на очередную машину.

Алкаш. Если вы в состоянии регулярно выдерживать фразы

типа «Я, б*, генеральный, б…, директор, б…, агентства, б…, по распространению, б…, культуры, б…», то в чистом виде этот тип для вас, пожалуй, самый безобидный из перечисленных, но дает страшную смесь с типами «гений» и «развлеканец». Обслуживание такого обычно ограничивается закупкой «самого дорогого компьютера», функционалом которого он не в состоянии воспользоваться, и обсуждением вопроса, нужно ли заказывать к самому дорогому компьютеру обшивку из дуба, вишни или красного дерева. Распоряжения алкаша, отданные в пьяном виде или с будуна, можно не выполнять без особых последствий для себя. Удручает, что вокруг алкаша обычно концентрируется много швали, необходимой ему в процессе выпивки и при выяснении вопроса «Ты меня уважаешь?», но шваль эта тоже опасна мало, поскольку пьяна большую часть времени.

Алкаш-развлеканец – пожалуй, один из наиболее опасных подвидов. Во всех кризисных ситуациях уходит в запой, оставляя компаньонов и менеджеров разбираться с навалившимися проблемами, а потом появляется в офисе с кучей новых идей и жгучим желанием их немедленно реализовать. Объявляется новый этап в развитии компании, меняются все правила учета, а поскольку новые правила никак не стыкуются со старыми, то долги по старым правилам амнистируются. Или не амнистируются, а конвертируются с помощью фразы:

– Будем считать, что он нам должен семь тысяч.

– А почему семь?

– Потому что я так решил.

> Есть еще любопытный типаж – «администратор». Как правило, увлекается построением сверхформальных процессов с кучей отчетности, формальным процессом согласований и большим количеством бюрократических уровней. С такими я часто встречался во внедренческих компаниях – в условиях, когда ни один проект не выполняется в срок с предсказуемым результатом, сверхформализация дает иллюзию того, что процесс управляем.
>
> К сожалению, побочным эффектом является обычно концентрация внимания на процессе в ущерб результату. *– Д. К.*

Кого-то из младших компаньонов объявляют виновными за приобретение неликвидных товаров, закупленных на этапе новой жизни двумя запоями ранее, и вычитают их стоимость из их паев. Офис трясет и лихорадит до начала следующего запоя.

Классификация главных главных бухгалтеров

В заголовке нет ошибки. Речь идет о людях, отвечающих за весь бухгалтерский учет в многоюрликой компании, у каждого юридического лица которой, естественно, есть отдельный главный бухгалтер. Я до сих пор не знаю, как таких назвать красиво. Может быть, генеральный бухгалтер.

Совсем главные бухгалтеры делятся на две большие категории. Первая категория разбирается в своей работе, а вторая повторяет «Хочу 1С». При главенстве второй категории учет проистекает как бы сам по себе либо определяется так называемым «1С-программистом», который разбирается в бухгалтерии лучше описываемого сотрудника. Удача программного продукта «1С: Бухгалтерия» привела к тому, что, даже достигая должностных вершин, очень многие продолжают верить, что учетная политика, план и корреспонденция счетов, а также типовые хозяйственные операции даны нам от Бога и изменяются тоже исключительно Божественным способом

Объяснить что-то хочуодинэсу практически невозможно: обычно они беспросветно дремучи. Я помню финансового директора, которому в течение получаса пытались объяснить, что умножение на 1,18 – это увеличение на 18 %. Когда это все-таки удалось, он, вздохнув, произнес: «Как у вас здесь все сложно. В 1С с таким разбираться не нужно».

Написанное совсем не означает, что нормальному главбуху не может нравиться 1С. Но он при этом понимает, что речь идет все-таки о средстве, которое в состоянии помочь бухгалтеру, но не заменить его.

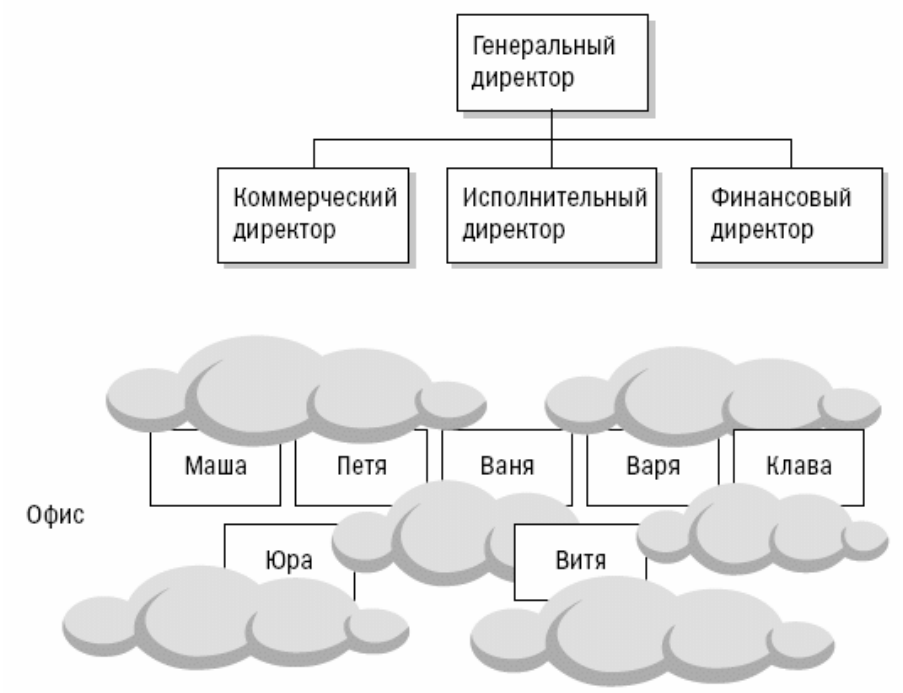
Затуманенные структуры управления

Изучение структуры управления компанией, в которой вы оказались, обычно является процессом, который не завершается вплоть до вашего увольнения. Иногда потому, что, поняв, как управляется компания, вы немедленно начинаете подыскивать новое место работы, иногда потому, что эта структура изменяется в том числе и вашими усилиями.

Знания в этой области многими руководителями воспринимаются как сокровенные, поэтому они достаточно часто скрываются не только от внешних наблюдателей (конкурентов, УБЭПа и т. п.), но и от собственных сотрудников. Так что неудивительно, что структура управления всегда видится вам в некотором тумане. И первое, что вас должно заинтересовать, – это напускают ли тумана конкретно вам или руководители сами плохо представляют, кто за что отвечает и кем руководит. К сожалению, чаще оказывается верным именно второй вариант,

хотя убеждать себя и вас будут в том, что речь идет о первом.

Один из примеров структуры (не худший) изображен на рисунке.



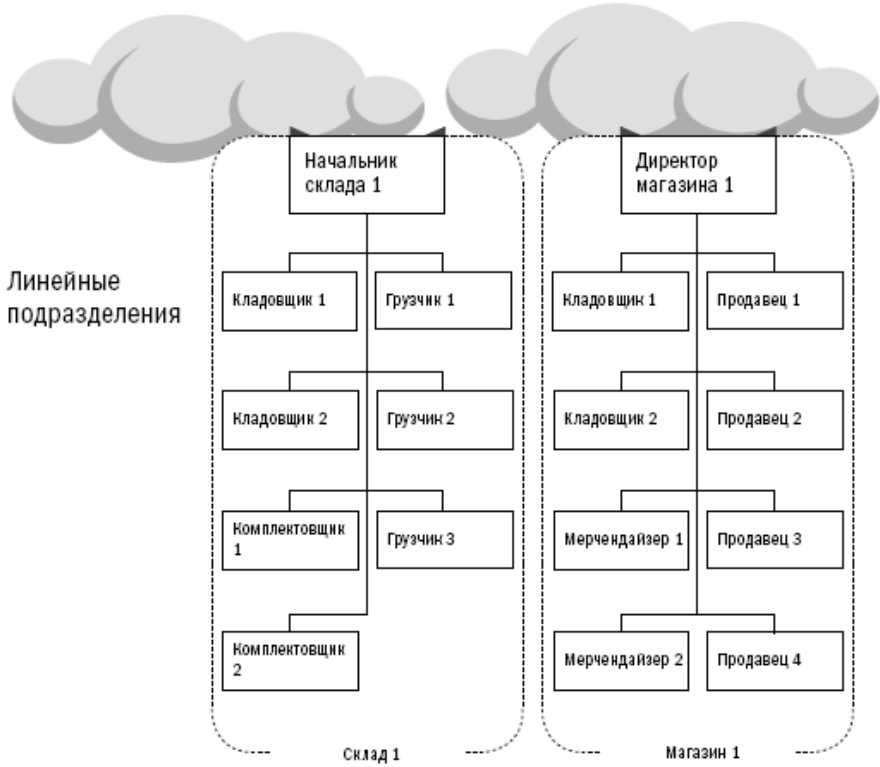


Рис. 1. Пример затуманенной организационной структуры компании

При такой структуре все подчиняются всем директорам, менеджеры офиса как-то договариваются друг с другом сами, а хорошо структурированные линейные подразделения подчиняются всем директивам офиса. Некоторые из офисных прямоугольников могут разворачиваться в подразделения (Маша может руководить бухгалтерией, а Ваня – айтишниками), но общего смысла это не меняет, поскольку способы взаимодействия подразделений все равно не заданы.

Затуманенные структуры обычно сохраняются у компаний, стартовавших с нуля, образовавшихся из группы единомышленников, объединенных одной бизнес-идеей. На начальном этапе организационная схема такой фирмы выглядит приблизительно так:

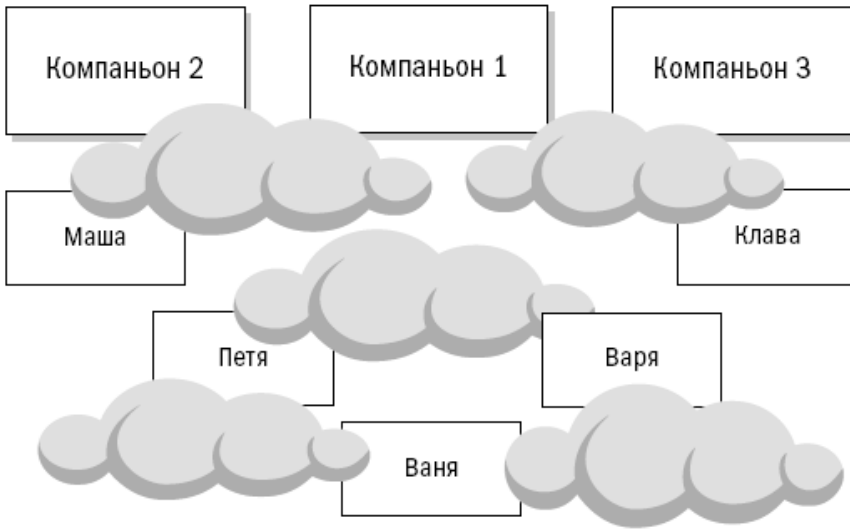


Рис. 2. Организационная схема свежеродившейся компании

На этом этапе развития в компании «все отвечают за все» и каждый выполняет любую работу, которую необходимо. Весь управленческий учет ведется, как правило, в Excel, а для бухгалтерского учета закупается или берется у знакомых коробочная программа.

В дальнейшем происходит постепенная структуризация сверху и снизу.

Нарастающие в нижней части схемы линейные подразделения обычно создаются уже структурированными. Может, и бывают случаи, когда этого не происходит, но тогда компания разоряется достаточно быстро, и до глобальной автоматизации дело не доходит.

Сверху постепенно происходит трансформация компаньонов: они отходят от бизнеса сами, съедаются, выживаются или разоряются до рядовых сотрудников. Реже компаньонам удается договориться о разделении функций и встроиться в традиционную структуру топ-менеджмента.

Если ничего из перечисленного не происходит, то схема превращается приблизительно в такого монстра.

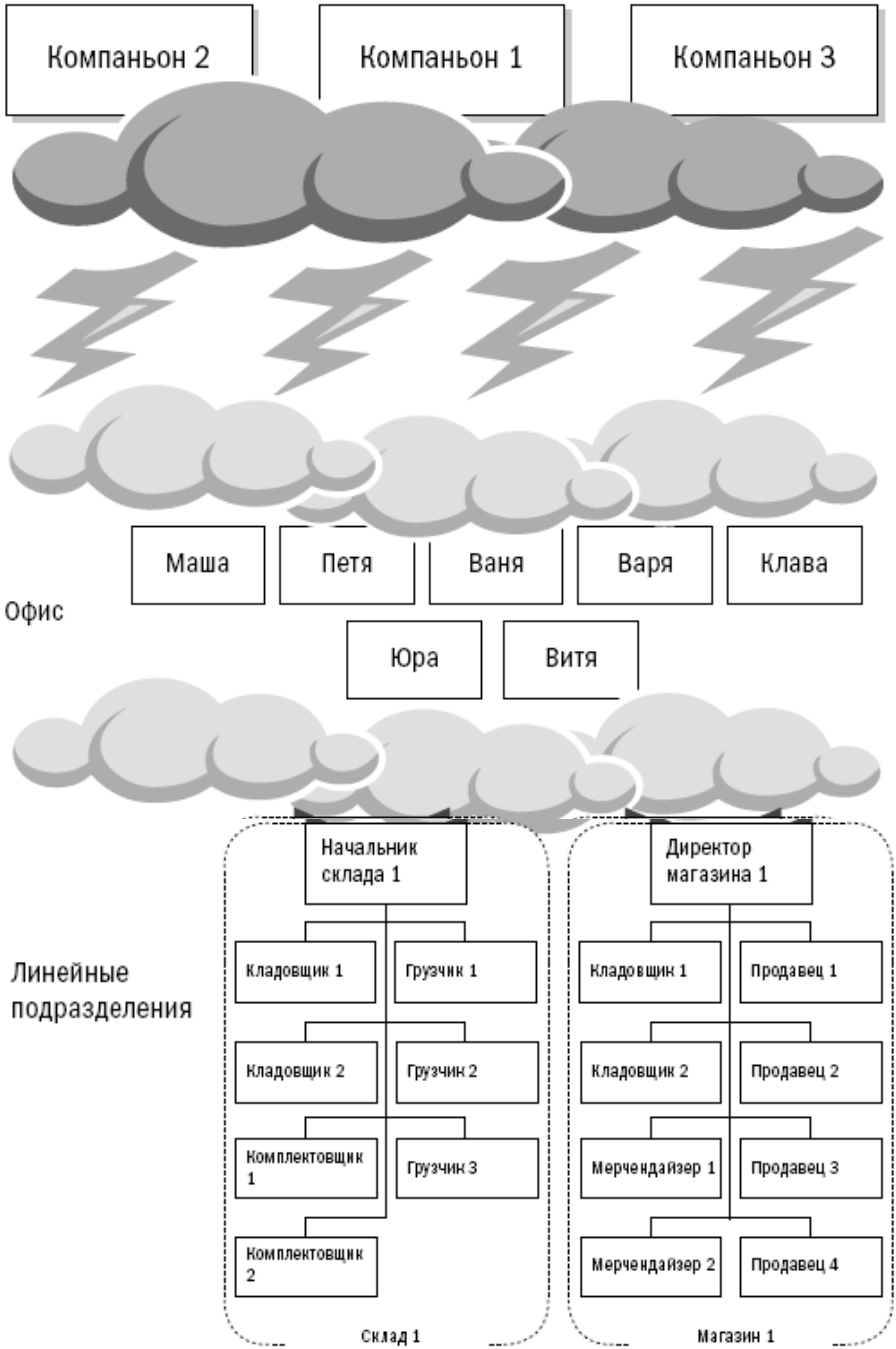


Рис. 3. Политеистическое управление компанией

Естественно, в таком виде компании долго не живут. Но автоматизаторы в такой структуре живут еще меньше.

Как ни странно, затуманенная структура «с головой», с которой я начал рассказ, бывает вполне работоспособной, пока не

разрастается слишком сильно. Мне встречался даже вариант структуры «все подчиняются троим, а дальше – как сами договорятся», который тоже работал. А называлось это почему-то «матричной структурой»

С другой стороны, всем известны организации с ясной классически-иерархической организационной схемой, жестким штатным расписанием, должностными инструкциями на каждую позицию, которые абсолютно мертвы и не способны ни к какой деятельности. Без внешней подпитки такие образования исчезают достаточно быстро, но, когда их питает государство или другой внешний источник, могут существовать веками.

К сожалению, процессы управления в компаниях с затуманенной структурой неформализуемы и потому плохо поддаются автоматизации. Если вы собрались внедрять только учетную систему, то основная проблема у вас будет выяснить, кто же в офисе является бизнес-заказчиком и с кем нужно согласовывать предлагаемые проектные решения.

«Этим занимается Ира»

Когда в компании используется затуманенная схема управления, вместо должностей в описании технологии работы часто используются имена. Конечно, во фразе «За это у нас отвечает Ира» особого криминала нет. Она может означать, что речь идет, к примеру, о функциях финансового контролера, которые сейчас исполняет Ира, а завтра, если потребуется, будет исполнять Маша.

Хуже, когда Ира оказывается неотделимой от своих функций по причинам:

а) высочайшего уровня доверия к ней;

б) уровня уникальных знаний и опыта;

в) отсутствия перечня выполняемых ею функций в виде, зафиксированном даже не на бумаге или в файле, но хотя бы в головах ее руководителей.

В результате при увольнении этой Иры выясняется, что те ее функции, про которые смогли вспомнить, необходимо разложить на двух других Ир, одну Машу, а также Костю, Пашу и Андрея, работающих вдобавок в совершенно других подразделениях. Людям этим нужно как-то согласовывать свою деятельность, чего в компании делать почти не умеют. А еще через пару месяцев, потеряв несколько миллионов рублей, руководство догадается, что у Иры были и еще какие-то обязанности, которые теперь не выполняет никто.

«Руководящая протоплазма»

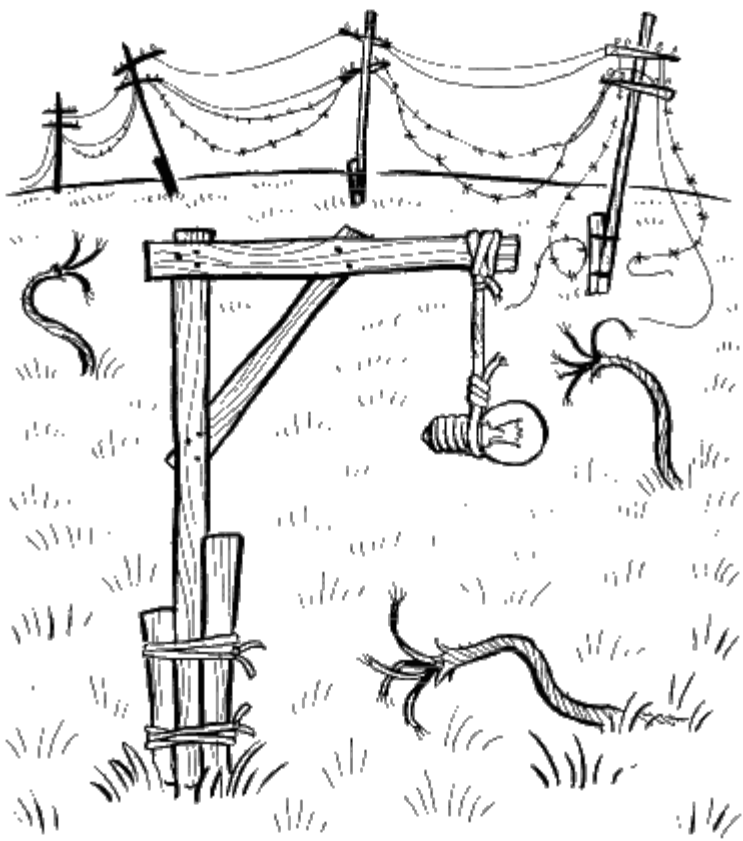
Чтобы было понятно, что даже в тумане бывают свои плюсы, приведу фрагмент из отчета об обследовании одного вполне успешного холдинга.

> Холдинг базируется на группе лиц, объединенных родственными или доверительными отношениями. Реально в этой группе каждый занимается тем, чем хочет и может, а совместные решения принимаются консенсусом, посему структура верхней части холдинга традиционно размыта и неформализуема. На мой взгляд, для ее описания больше всего подходит термин «руководящая протоплазма» (я надеюсь, что у заказчика хватит ума не обижаться). За каждым из членов руководящей протоплазмы закреплено некоторое число сотрудников, которыми он распоряжается и функции которых уже более четко определены. Впрочем, ни закрепление, ни сами функции обычно никаким способом не зафиксированы и не формализованы, а посему тоже размыты.
>
> Понятие должности проявляется только при печати визитных карточек или заполнении трудовых книжек, должностные инструкции формулируются устно и поддерживаются лишь на уровне их интуитивного понимания исполнителем.
>
> Описанная модель работает уже достаточно долго и эффективно, так что глупо обсуждать ее правильность или неправильность. Шок возникает лишь у наемных сотрудников, в первый день работы получающих вводную: «Подчиняться будешь А (эта часть иногда опускается), выполнять задания Б, В и Г, а зарплату получать у Е». Впрочем, этот шок проходит, когда сотрудник убеждается, что у него получается и работать, и получать зарплату.
>
> Механизм реализации холдингом бизнес-проектов практически неизменен. Сначала возникает бизнес-идея. Для ее реализации выбирается доверенное лицо, которому делается соответствующее предложение. Это лицо вводится в руководящую протоплазму (если оно ранее в ней не находилось). После этого выбранное лицо создает бизнес-план, обсуждает его в протоплазме и начинает его реализовывать, постепенно формируя вокруг себя ядро сотрудников, занятых проектом. В какой-то момент проект либо коллапсирует, либо выделяется из протоплазмы в виде отдельного

подразделения или организации, существующей практически независимо и связанной с холдингом только финансово.

Как следствие, внутри холдинга постоянно существует некоторое количество проектов на эмбриональном и постэмбриональном уровне, что дополнительно затрудняет попытки формализованного описания его функционирования.

Разберитесь



Как вы все-таки сможете понять, чего же от вас хотят, я не знаю. Знаю только, что на это уйдет уйма времени, сил и нервов, но результат все равно будет плачевным.

Способность начальников объяснить свои информационные потребности за последние тридцать лет изменилась мало: если раньше они просто размахивали руками, то теперь делают то же самое, но растопырив пальцы.

Что интересно, предшествующее руководящей работе образование на их мыслях и поступках практически не сказывается: и прикладные математики, и теоретические бухгалтеры говорят и поступают одинаково.

(Наверное, сейчас я должен извиниться перед руководителями, дочитавшими мои записки до этого места. По всей видимости, вы – приятное исключение из общего печального правила. И я даже иногда встречал таких и с удовольствием с ними работал. Но как же часто после начальственных бесед и производственных совещаний я вспоминаю сцену из кинофильма «Бумбараш», в которой бандит, стреляя в направлении художника, кричит: «Нарисуй мне кунгуру!»)

Ниже приведены фрагменты отчета об обследовании еще одной корпорации, в которой я оставил несколько лет своей жизни.

Памятная записка «Проблемы автоматизации компании»

<…>

1.4. Задачи автоматизации

1.4.1. В качестве идеала высказывается желание иметь в центральном офисе всю информацию о закупках товаров, их хранении и продажах во всех торговых точках, складах и офисах, обо всех перемещениях финансовых и материальных средств, а также обо всем остальном в атомарном (не интегрированном) виде.

1.4.2. Ограничения по срокам фактически отсутствуют (поскольку все должно было работать вчера), ограничения по средствам также не определяются, хотя, по всей видимости, существуют.

1.4.3. Оценки эффективности той или иной информации для корпорации (убытки из-за отсутствия информации, прибыли от ее наличия) не проводились или недоступны.

<…>

1.6. Проблемы автоматизации

1.6.1. Рабочие места. Отсутствие рабочих мест суммарной стоимостью не более $2000 на человека для специально нанятых высокооплачиваемых сотрудников удивляет. Очевидно, что производительность труда человека, не имеющего даже собственного стула, не может быть высокой при всем его желании работать, которое тоже быстро исчезает.

1.6.2. Интерфейсы. Получение информации и общение с руководством происходит через «игольное ушко» – постоянно загруженного финансового директора. По всей видимости, этот документ тоже осядет в его портфеле. Доступ к сотрудникам корпорации для получения информации напрямую затруднен в связи с загруженностью последних и, может быть, из соображений секретности.

> Почему бы и нет? На одном из крупных предприятий мне отказались предоставить формы счета-фактуры, объясняя это соображениями секретности.
>
> На другом после обследования консультантов просили вернуть все копии документов (отдельно оговорив«„включая электронные копии“«).
>
> Секретность – вообще область интересная. Мне рассказывали, что одну девушку просили после перемены фамилии заново пройти инструктаж по безопасности. – *Д. К.*

1.6.3. Структура. Неопределенность статуса сотрудников, занятых автоматизацией, внутри фирмы, отсутствие определенных каким-либо способом должностных обязанностей мешают в любой области деятельности, а при построении систем управления это просто анекдот. («Врачу, исцелися сам»)

1.6.4. Финансовые ресурсы. Никому не приходит в голову выбирать себе автомашину «вообще» – каждый привязывает этот процесс к своему кошельку. Некоторые останавливаются на городском транспорте. У меня есть подозрение, что я оптимизирую выбор между «Роллс-Ройсом» и шестисотым «Мерседесом», когда денег хватит в лучшем случае на «девятку».

1.6.5. Сроки. Отсутствие реальных сроков выполнения работ, даже завуалированное фразой, что все нужно было сделать вчера, развращает и исполнителей, которые в любом случае не могут выполнить работу в срок, и руководителей, которые не заботятся о своевременном выделении ресурсов.

<…>

Теперь вы поняли, что если вы работаете в таких условиях, то здесь я вам не помощник. Дерзайте сами. Ведь именно вы – «высококлассный профессионал-постановщик».

Но предположим, что вы все-таки догадались, что вам нужно сделать.

Очень много сил уйдет у вас на попытки объяснить руководству, что в учете не надо изобретать велосипед.

«Зачастую исполнительный директор страшно горд тем, что придумал велосипед с шестиугольными колесами вместо использовавшегося им ранее велосипеда с квадратными колесами. Он и не подозревает, что у нормальных людей колеса давно круглые». Эту фразу я услышал в 1996 году от А. В. Ширченко. К сожалению, я попытался записать ее по памяти только лет через пять, и поэтому цитата может быть неточной.

Почему-то особенно тяжело доходит, что бухгалтерский учет итальянцы изобрели еще в XV веке для того, чтобы считать свои деньги, а вовсе не для того, чтобы морочить голову государству. Поэтому и ныне его можно эффективно использовать для той же цели.

Еще интереснее, если ругательства используются, но вот смысл в них вкладывается совсем другой. Например, мало кому можно объяснить, что «центр затрат» – это совсем не синоним выражения «подразделение, не приносящее дохода». *– Д. К.*

Страшно начальникам, не отличающим двойную запись от двойной бухгалтерии. Про водку они обычно знают все, а на бухгалтерскую проводку они и взглянуть боятся. Может быть, у вас что-нибудь получится, если вы клятвенно заверите руководство, что, несмотря на использование стандартных бухгалтерских методик и систем в отчетах, с которыми будут работать они, не будут встречаться бухгалтерские «ругательства» (типа баланс, актив, пассив, дебетовое сальдо и кредитовый оборот по субсчету)

Научитесь

Если, прочитав предыдущую часть записок, вы надулись от гордости за себя и свою профессию, то теперь самое время сдуться обратно. Потому что настало время учиться.

Все фирмы функционируют приблизительно одинаково? Вы все и так хорошо знаете? Тогда скажите, какая разница существует между терминами «заказчик» и «клиент»?

Если не знаете, то вам, наверно, рано писать задание для автоматизации учета в морге или крематории.

Простенький вопрос: может ли уменьшение

розничной цены привести к уменьшению объема продаж в штуках? Нет? Вы уверены?

Ответ. При внедрении информационной системы в компании, владевшей газетными киосками в метро, киоскеры стали жаловаться на снижение продаж из-за того, что цены на газеты были заданы с точностью до копеек: возня со сдачей приводила к формированию очереди, и покупатели уходили в соседний киоск, где те же газеты стоили дороже, но не было очереди. Цены округлили до полтинника в большую сторону, и количество продаж увеличилось.

К сожалению (или к счастью – как для кого), чтобы разработать хорошую систему, вам следует быть не только прекрасным знатоком всех областей ИТ – вы еще должны разбираться в работе каждого подразделения своей фирмы не хуже, а лучше руководителя этого подразделения. Только тогда предлагаемые вами решения будут если не оптимальными, то хотя бы работоспособными.

Еще один пример. У фирмы, на которую вы работаете, есть два магазина: в центре Москвы и у кольцевой дороги вокруг Урюпинска. Соответствующий менеджер убеждает вас, что цены в этих магазинах одинаковы, всегда были одинаковы и будут одинаковы вечно.

Вы должны понимать, что в этом случае вечность может закончиться через неделю, и если:

– цены в магазинах не будут выравниваться системой автоматически до окончания вечности;

– цены в магазинах не смогут стать разными сразу после ее окончания, то вы спроектировали систему плохо.

Одновременно с изучением надо тем или иным способом описать существующую технологию работы фирмы. Это очень нудно, но необходимо. Тому, что это делать необходимо, меня учили в институте еще при советской власти, тридцать лет назад. Технологию редко кто описывал по-человечески раньше, ее редко описывают нормально и сейчас. Причиной тому служит кажущаяся бессмысленность этой работы: старая технология после описания сразу же изменяется, а новая всегда тоже отличается от описанного в проекте варианта. Да и средств отразить все нюансы технологии не существует.

Тем не менее я продолжаю считать не только полезным, но и

необходимым скрупулезное описание технологии функционирования организации как на этапе обследования, так и на этапе проектирования.

Утешением вам может служить то, что на этом этапе вы в первый раз можете принести существенную пользу своей фирме (правда, я не уверен, что это оценят. Не исключено, что, напротив, скажут: «Сует нос куда не надо»). Следующий раз будет только после внедрения, а будет ли еще внедрение…

В качестве средства описания мне до сих пор нравятся старые добрые структурно-информационные временные схемы (СИВС), так как на них даже руководству видно, через какое место организован документооборот, но используемые средства определяют скорее эстетику документации. А вот если вы не можете описать технологию никак, это, очевидно, характеризует или вас, или фирму.

В отзывах на первую интернет-публикацию записок достаточно часто спрашивали, где можно почитать про СИВСы. Все поисковые сервера Интернета при попытке что-нибудь о них узнать, дружно отправляют меня к моим же запискам. Похоже, что вся информация так и осталась в 1970-х…

> Это не так. Сейчас это именуется dataflow diagram (или audit diagram, или еще как угодно в зависимости от рассказчика) и используется при описаниях процессов. Один из примеров СИВС-подобной нотации – ARIS EEPC, используемая в одноименном продукте (к сожалению, сильно недешевом). *– Д. К.*

СИВСом называлась диаграмма, по оси абсцисс которой откладывалось время, а по оси ординат – организации, структурные подразделения или сотрудники. Когда меня учили, на СИВСе отображали документы и действия с ними. Я размещаю на СИВСе все объекты, которые существенны в бизнес-процессе, включая пачки денег и товары, если это необходимо. Получается очень наглядно и понятно даже неспециалисту. Не слишком хорошо удается отобразить только циклы. Для их отображения я просто выделяю участок диаграммы, под которым пишу текстом «Повторяется 10 раз» или «Повторяется, пока все не согласятся с текстом»

В последние годы для рисования СИВСов я использую Microsoft Visio. Пример приведен ниже.

> Да, вполне достаточно любой рисовалки, хоть

текстового редактора (даже лучше не увлекаться дорогими продуктами, так как смысл работы может утонуть в процессе увлекательного освоения ультрамодной технологии). – *Д. К.*

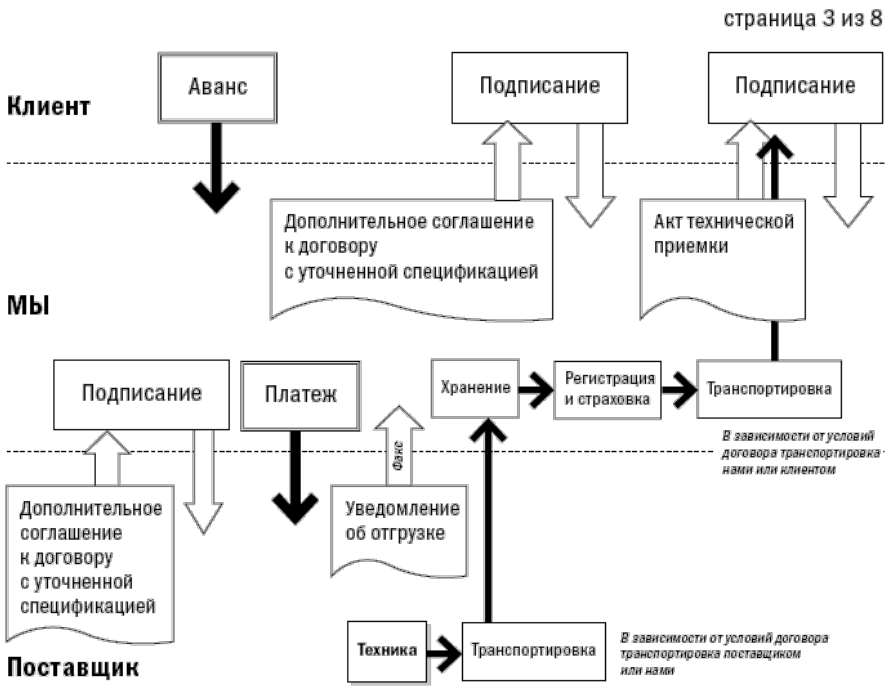


Рис. 5. Пример фрагмента СИВС

Следует отметить, что СИВС в моей интерпретации не предполагает автоматической генерации программ и баз данных. Эта штука принципиально недоформализована, что позволяет ее делать настолько наглядной, насколько это нужно. Если же вы хотите автоматизировать всю разработку от описания бизнес-процессов до кодов, то нужно использовать те средства поддержки case-технологии, которые сейчас есть на рынке.

Окопайтесь

Помимо осознания информационных потребностей компании вам еще очень неплохо сформулировать основные задачи ИТ-подразделений. Не надейтесь, что ваше мнение на этот счет совпадает с мнением руководства. На самом деле подразделения ИТ не только могут работать, но и на самом деле работают, руководствуясь разными целями, и единодушие среди топ-менеджмента хотя бы в этих вопросах будет для вас совсем не

лишним. Разберем пример документа.

Концепция управления подразделениями ИТ

1. Направление ИТ состоит из подразделений, занятых в основном обслуживанием других подразделений компании, обеспечивающих поступление прибыли. Сами подразделения ИТ планово затратны и не ориентированы на получение прибыли с помощью продажи оборудования, программ или услуг.

Это означает, что при проектировании программного обеспечения и проработке технологических решений подразделения не закладываются средства на возможность дальнейшей продажи решений, поскольку последнее требует существенно больших ресурсов в процессе разработки. С другой стороны, топ-менеджмент компании не рассчитывает на продажу этих решений и не соблазняет такими мыслями сотрудников ИТ-подразделений. Следует помнить, что предпродажная подготовка программного продукта может составлять более девяноста процентов его стоимости.

2. Сотрудники ИТ-подразделений могут иметь и высказывать свое мнение по вопросам организации торговли, развития фирмы и другим вопросам, находящимся вне зоны их основной специализации, но всегда выполняют решения профильных менеджеров компании, какими бы странными они им ни казались.

Как мне кажется, для этого пункта альтернативы не существует. То есть она есть, но ровно одна: айтишник становится исполнительным директором или кем-то похожим. Но в этом случае он перестает быть айтишником…

Как следствие, инициативы создания и развития программных продуктов пресекаются, если они не одобрены топ-менеджментом компании.

3. Существуют приоритеты в очередности использования ресурсов для решения задач подразделениями ИТ (представлены в порядке убывания значимости для компании, в которой вы работаете):

– Обеспечение возможности принятия стратегических решений топ-менеджментом компании (организация учета и отчетности, позволяющей принимать решения);

– Увеличение количества продаж (изменения в ПО для проведения торговых акций) и скорости продаж в магазинах (изменения ПО для облегчения и ускорения работы в кассовых программах);

– Обеспечение роста производительности и удобства работы других пользователей.

Полезно также помнить, что автоматизация учета должна предшествовать автоматизации планирования. Почему-то иногда пытаются сделать наоборот. – *Д. К.*

Тоже вроде все ясно. Но хочется, чтобы все желания исполнялись сразу. Я в таких случаях цитирую концовку сказки Сергея Михалкова «Жадный Вартан»: «Больших семь шапок из овцы / Не выкроишь никак!»

Есть три сказки, с помощью которых удобно иллюстрировать процесс постановки задачи и последующего внедрения.

1. «Жадный Вартан» (когда урезанный бюджет приводит к сильному сужению рамок проекта, что иногда выливается во внедрение журнала хозопераций вместо полноценной системы).

2. «Каша из топора» (когда заключается договор с узкими рамками, которые впоследствии вместе с бюджетом плавно расширяются внедренцем).

3. «Сытный бублик» (когда результат достигается только после многочисленных недовнедрений, в процессе которых набирается опыт и появляется понимание того, какой все-таки был необходим результат). Последняя попытка в таком случае оказывается удачной, и руководитель восклицает, подобно деду из сказки: «Что же ты мне сразу этот сытный бублик не дала?!» – *Д. К.*

4. Предложения по доработкам и развитию ПО и оборудования могут выдвигать и обсуждать все сотрудники компании, однако выполнению подлежат только заявки, утвержденные руководителями направлений уровня заместителя директора.

Вас удивляет, что это надо писать? Вот поверьте, надо.

И принимайтесь за работу

Теперь вам придется определиться, воспользоваться покупной системой или программировать самим. И то и другое имеет свои очевидные минусы: покупные системы никогда не заточены под ваши задачи, а на собственную требуется такое количество ресурсов (и *денег*, и *людей*, нанимаемых не на постоянную работу, а «под задачу» – что, бесспорно, ускоряет решение этой задачи, – и, что самое важное, *времени*), какое в большинстве случаев просто недоступно.

Что касается собственной разработки, то можно почти с уверенностью сказать, что седло у велосипеда, который изобретете вы, будет удобно для зада вашего хозяина. Не вполне ясно только, будет ли этот велосипед ездить. И чем крупнее и многопрофильнее компания, тем меньше шансов, что такая работа когда-нибудь завершится. Впрочем, и в последние годы я видел завершенные проекты. Но мало.

Если вы остановились на покупной системе, то ее, к сожалению, нужно выбрать. Выбор богатый, и все варианты вам, конечно, не нравятся. Но кажется, тут я вам могу немного помочь.

Выбор системы

Вот уж в чем сейчас недостатка нет, так это в статьях по выбору тиражируемой информационной системы и компании-внедренца. Появились даже фирмы, специализирующиеся на консультационных услугах по выбору тиражируемых

Такие уже есть. – *Д. К.*

информационных систем. Подозреваю, что скоро появятся фирмы, специализирующиеся на выборе таких консультантов. Кстати, среди статей встречаются вполне разумные. К сожалению, во многих других случаях при чтении статьи необходимо держать в голове, где работает автор и продажу каких систем проталкивает.

Ангажированность авторов, изучающих и обозревающих эти вопросы, зачастую видна уже на уровне определений и классификации. Очень много сил отдано проработке определения КИС – корпоративной информационной системы, ИУС – информационно-управляющей системы, ИСУП – интегрированной

системы управления предприятием, системы класса ERP – Enterprise Resource Planning, ERPII, стандарта MRPII – Manufacturing Resource Planning. После того как определение тем или иным способом сформулировано, обычно делается вывод, что информационная система, продвигаемая авторами публикации, определению соответствует, в отличие от систем-конкурентов.

> Да, это все равно что говорить, что продукт соответствует стандарту ISO 9000 (видел я и такое). – *Д. К.*

При этом ERP-стандарта попросту не существует, а MRPII американской ассоциации управления производственными ресурсами APICS – это концепция управления производством и запасами, то есть методология менеджмента, а не стандарт информационной системы.

Когда вы подбираете систему, вам не очень интересно знать, является ли она настоящей полнофункциональной кисой в понимании эксперта NN. Почему какая-то функциональность называется полной? Полной для какого предприятия? Как должна влиять на мой выбор информация о том, что система удовлетворяет информационные потребности девяти из десяти организаций, если я работаю как раз для десятой? И почему я в этом случае должен платить за систему дороже? Так что вам в любом случае придется сравнивать перечень нужных вам функций с перечнем функций рассматриваемой системы. Причем по названиям функций вы ничего не поймете, названия у всех разработчиков поворачиваются по веянию моды, как флюгер, гораздо быстрее, чем в систему вносятся необходимые изменения. В какой системе сейчас нет логистики, бюджетирования, МСФО? И как вы удивитесь, когда поймете, что имел в виду разработчик… Но об этом ниже.

> Тем более удивляет огромное количество публикаций консалтинговых компаний, проводящих сравнение программ по их функциям. Читая такие обзоры, часто понимаешь, что сделаны они именно по названиям функций. На такие рейтинги и обзоры часто любят ссылаться продавцы систем («Наша система набрала 99 очков из 100 по шкале компании Ы» или «По оценке экспертов N наша система является лидером в области Э»). Встречаются даже совсем курьезные тексты – например, описания того, что «система N повышает удовлетворенность пользователей на 30 %», или попытки

сравнивать программы на основании получаемого разработчиком дохода от продаж. – *Д. К.*

Тем не менее некоторые характеристики, очевидно, служат для отбора систем до их подробного изучения. К таковым относятся:

– максимальное количество рабочих мест, на которое рассчитана система;

– оценочная стоимость и продолжительность внедрения;

– место разработки системы: зарубежная или отечественная;

– возможность дорабатывать систему под нужды клиента (открытость системы).

Максимальное количество рабочих мест

Этот показатель позволяет сразу отсечь маломощные системы, созданные с использованием устаревших программных средств общего назначения (СУБД и операционных систем) и устаревших технологий. Кажется разумным отсекать те системы, максимальное количество рабочих мест в которых меньше удвоенного, а еще лучше утроенного количества рабочих мест, закладываемых в проект. Понятно, что даже если вы информацию о количестве рабочих мест получаете у разработчика, он должен дать вам адрес и телефон организации, где система на этом количестве рабочих мест функционирует.

Техническими параметрами системы в ходе внедрения управлять гораздо сложнее, чем бизнес-логикой (трудно заставить систему, рассчитанную на работу двадцати пользователей, обслуживать пять тысяч операторов). Поэтому именно на технические параметры стоит обращать больше внимания. А рассматривая технические характеристики, не лишне также заранее подумать, как же будет работать система, если необходимый для ее функционирования ресурс, воспринимаемый разработчиком как навечно данный от Бога, вдруг перестанет быть таковым (например, предположение о том, что каналы связи от Москвы до подразделения в Магадане вдруг могут перестать работать, может привести к выбору системы, поддерживающей работу с распределенной базой данных).

Значение технических характеристик существенно еще и потому, что бизнес-логику одной системы гораздо

легче скопировать в другой (в результате чего во многих системах, особенно западных, заложены практически идентичные процессы). Быстродействие и возможность поддержки больших объемов данных из удачной системы скопировать гораздо труднее.

Оценивая производительность системы, нужно также обращать внимание и на то, какая функциональность использовалась пресловутыми тысячами пользователей в компании N. Хотя факт запуска на огромных объемах данных или на большом количестве рабочих мест и свидетельствует о возможностях системы, но совсем не гарантирует, что именно локализация для вашей страны или решение для вашей отрасли будут работать. – *Д. К.*

Оценочная стоимость и продолжительность внедрения

В большинстве случаев компании, осуществляющие внедрение информационных систем, стараются заранее занизить оба показателя в надежде, что клиент доплатит и подождет, оказавшись с неработающей системы после того, как закончились и выделенные деньги, и выделенное время. С другой стороны, после получения первых порций денег внедренцы всегда в состоянии довести оба показателя до любой величины, если заказчик не разорится раньше.

Реальным критерием является лишь минимальная величина по полномасштабным осуществленным проектам. Сейчас такие оценки достаточно часто встречаются в обзорных статьях.

Только поймите, это не те сумма и время, которые потратите вы. Эта те сумма и время, меньше которых вы не потратите точно.

Настройки и коды

Любую тиражируемую систему нужно настраивать для конкретной организации. Это ни у кого не вызывает возражений. Точно так же не вызывает возражений, что настройка лучше дописывания или изменения программных кодов. Вот только что считать настройкой, а что кодами?

Пока нужно ставить галочки в табличках, выбирать какие-то значения из списка и даже вписывать какие-то числа, это точно настройка.

Пока пишешь формулы для расчета цен и сумм проводок в типовых операциях, это, наверное, тоже настройка.

Если таким формулам предшествуют условия, при которых они применяются, то я готов и это признать настройкой.

Но дальше – больше, потребности растут, и вот разработчик добавляет в систему свой корявый алгоритмический язык и обзывает тексты на нем тоже настройкой.

Осмысленность последнего для меня очень сомнительна. С одной стороны, обычно у разработчика информационной системы нет специалистов для создания хороших интерпретаторов и трансляторов, так что результат получается чрезвычайно убогим, а с другой – основная сложность в программировании на известных алгоритмических языках состоит отнюдь не в том, что используемые ключевые слова написаны по-английски. Человек либо может описать алгоритм, и тогда в течение недели сделает это даже на языке с ключевыми словами на суахили, или не может, но тогда его не спасет то, что вместо «go to» нужно писать «идти на».

Для меня самого разница между настройкой и программированием, к примеру, для модулей бухучета определяется очень просто: если такую работу может сделать квалифицированный бухгалтер, то это настройка. Если для этого нужен программист, то это программирование. Другое дело, что «настроечное» программирование не должно задевать ядра системы, но для этого давным-давно разные способы придуманы.

К настроечной части системы также принадлежат генераторы экранных и печатных форм и генераторы отчетов. Они очень облегчают жизнь, позволяют использовать на сопровождении часть специалистов более низкого уровня квалификации. Только моя практика показывает, что с их помощью удается легко решать только простые задачи, а в сложных случаях все или зависает, или начинает работать очень-очень медленно.

Российский самопал или крутой Запад?

Да, в западные системы денег обычно вбухано гораздо больше, чем в российские (только вот в разработку или в пиар?). Правда, надо учесть, что и программист российский стоил до недавнего времени в пять раз дешевле аналогичного западного.

> Одной из мантр поставщиков ERP является утверждение, что на разработку системы потрачены тысячи человеко-лет. И действительно, в компаниях, поставляющих крупнейшие ERP-системы, работает очень

и очень много квалифицированных специалистов. Однако следует обратить внимание на следующие моменты:

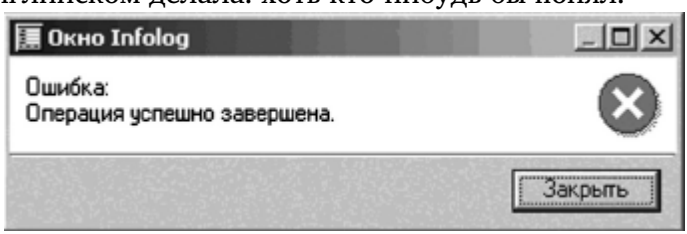
– большинство этих специалистов работает вовсе не над теми проблемами, которые стоят перед вами;

– численность людей, занятых локализацией продукта, вполне может быть сопоставима с численностью специалистов, занятых поддержкой этого продукта у заказчика;

– огромное количество этих человеко-лет потрачено на функциональность, которая никогда на данном конкретном предприятии использоваться не будет;

– попытка доработать тиражную систему может оказаться гораздо сложнее разработки с нуля, так как поневоле придется поддерживать часть ненужной на конкретном проекте функциональности и бороться с чужим программным кодом (это, кстати, также одна из причин того, почему заказная разработка всегда будет работать на порядок быстрее). *—Д.К.*

Да, количество внедрений конкретных зарубежных систем на порядки выше, чем российских (если, конечно, не считать 1С), поэтому системы более отлажены. Но это не касается их российской локализации, которая, кстати, делается российскими программистами. Посему не удивляйтесь, если в какой-то момент система в задумчивости начнет разговаривать с пользователями на английском языке или на таком русском, что уж лучше бы она это на английском делала: хоть кто-нибудь бы понял.



То, каким образом система поддерживает российское законодательство и российские обычаи, тоже следует изучать очень подробно и скрупулезно.

Самая крупная для нас неприятность в зарубежных системах состоит в том, что они обычно ориентированы на наличие общего стандартного регламента управления деятельностью предприятий, который пока почти не применяется в России. Посему компания, осуществляющая внедрение, вместо настройки системы под существующие в организации бизнес-процессы в большинстве случаев предлагает выполнить «реинжиниринг»

бизнес-процессов, чтобы они соответствовали бизнес-процессам и технологиям, предусмотренным в данной системе, то есть перестроить всю технологию работы организации.

Если уж вы выбираете импортную систему, то вряд ли у вас получится просто создать накладную на отпуск товара, если товар не оприходован на складе. В отечественных системах контроль ухода остатков в минус обычно можно включить или выключить, и везде, где я работал, меня просили этот контроль выключить.

Чтобы понять некоторые неудобства от работы с такой системой, вспомните свой опыт работы с MS Word, который всегда лучше вас знает, как форматировать текст, нумеровать и оформлять списки, учит вас, что в русском языке прилагательные с приставкой «не» всегда пишутся отдельно, и предлагает исправить выражение «номинировать на конкурс» на правильное: «но минировать на конкурс». Если вас это все не бесит, вы, может быть, и с иностранной системой уживетесь.

Изучая ИХ системы с точки зрения бухгалтерского учета, понимаешь, как ИМ ТАМ хорошо живется: в типовых настройках одной экономической транзакции соответствует одна бухгалтерская проводка. И еще одна на НДС.

– А больше можно? Ведь нам даже в официальном учете их необходимо сделать пять.

– Можно, если скриптик написать.

Я уж не говорю про желание вести параллельно «управленческий» и «официальный бухгалтерский» учет. Да нет, я даже не про черный и белый. Я про желание вести учет по какому-нибудь международному стандарту, чтобы его зарубежный инвестор понимал, и по российскому, чтобы перед налоговой отчитываться. Не понимают этого ни люди иностранные, ни системы: учет может быть только один.

> Еще раз хочу напомнить о том, что большинство зарубежных систем функционально мало отличаются, так как разрабатывались с прицелом на одну методологию. Посему разница между ними часто не в функциональности, предлагаемой пользователю, а в используемой технологии и производительности (что и должно, наверное, являться ключевым при выборе). – *Д. К.*

Результатом этой ситуации становится то, что некоторые на

полном серьезе даже предлагают считать нормальным «совместное использование зарубежной системы класса ERP с российской системой бухучета», как я читал в одной статье. Сразу видно, что автор эту зарубежную систему класса ERP хотел кому-то продать, а не эксплуатировать.

Конечно, любую систему можно доработать до состояния, когда она вас удовлетворит (например, переписав заново все, кроме логотипа). Но на мой взгляд, учитывая, что зарубежная система обойдется вам в $2-10$ раз дороже аналогичных российских, выбор таких систем сейчас должен иметь очень существенные основания, например, наличие функциональности, которой нет в отечественных системах. Но если этого требует ваш зарубежный совладелец, куда же вы денетесь…

Кстати, о логотипе. Не забывайте сами и не уставайте напоминать руководству, что корпорации Microsoft Inc. не разрабатывала систем Navision и Axapta[1]

Она купила фирму Navision вместе с ее программными продуктами,[2] поэтому ждать от этих продуктов исключительной совместимости с продуктами Microsoft не нужно. Равно как не нужно надеяться, что при разработке этих продуктов использовались соответствующие технологические стандарты корпорации Microsoft. Компания SAP пошла еще дальше: она купила продукт, который сейчас продает под названием SAP Business One, у израильских разработчиков, не купив при этом самих израильских разработчиков, так что самостоятельно не может даже исправить ошибки в этом продукте. Настоящие «Жигули» с официально установленной трехлучевой звездой от «Мерседеса».

Характеристики разработчиков и внедренцев

Представьте, что вы нашли ошибку в продукте мелко-мягкой корпорации. Думаю, что с вами это уже не раз происходило. Может быть, вам сложнее представить, что у вас есть оплаченная и зарегистрированная лицензия на продукт, в котором вы нашли ошибку, но постарайтесь представить и такое. Получилось? А теперь оцените вероятность, что эту ошибку исправят. Ну как, это число сильно от нуля отличается? А куда отправить ваши пожелания по расширению функционала этого продукта, вы тоже догадались? Это я про то, что кроме крутости бренда и широты функционала вам никогда не следует забывать и о том, насколько вы в состоянии повлиять на развитие продукта или хотя бы исправление ошибок в нем.

К сожалению, как бы вы ни старались, плотно познакомиться с выбираемой системой удастся только в процессе ее внедрения. Результаты этого знакомства иногда оказываются ошеломляющими. Поэтому при выборе системы помимо изучения характеристик ее самой нужно попытаться получить как можно больше информации, характеризующей ее опосредованно. Очень многое о своей собственной дальнейшей судьбе можно узнать, поняв, как устроены фирма-разработчик и фирма-внедренец. Это касается как основополагающих, базовых документов, определяющих стратегию, принципы и дисциплину разработки и внедрения, так и структуры фирм, персональных данных ключевых фигур. Даже биографией генерального директора стоит поинтересоваться. И если окажется, что его выгоняли из института за пофигизм, то подумайте хорошенько: это свойство не выветривается.

Документы разработчика: стандарт предприятия на разработку программного обеспечения. Документ не всегда называется так, но включает общие правила разработки программ и их объединения в программный продукт. Обратите внимание на оформление модулей и объектов (ограничения на размер, титульные комментарии, правила разбиения на строки, количество пробелов в отступах, подробность комментариев, способы внесения изменений, основные принципы взаимодействия модулей, как можно вызывать и передавать параметры, основные правила коллективной работы над проектом). Стандарт предприятия почти нигде и никогда не выдерживается строго, но его полное отсутствие означает такой подход к разработке, который в любом случае принесет вам неисчислимые беды.

Документы разработчика: модель базы данных. Опять же не встречалась мне система, структура данных которой в точности соответствовала бы модели, которую мне показывали: в процессе доработки продукта структуру данных всегда меняли, а вносить изменения в модель было уже лень. Но не это самое страшное. Гораздо хуже, когда в процессе разработки модель вовсе не рисовали, а разработчики даже не понимают, зачем она нужна. Это может означать, что последние пятьдесят лет опыта разработки информационных систем прошли мимо разработчиков и в процессе внедрения эти изобретатели велосипеда наступят на все грабли, отполированные лбами их предшественников.

Как истинный ретроград, я предпочитаю смотреть на модель «сущность – связь» (ER-модель), но годится и любая другая общеизвестная модель, лишь бы она была описана в литературе, а не придумана самими разработчиками.

Документы внедренца: стандарты предприятия на предпроектное обследование организации и внедрение информационной системы. Как и в предыдущих случаях, главное, чтобы документы были. Думаю, по их прочтении вы легко поймете и интеллектуальный уровень компании, с которой вы собираетесь связаться, и ее багаж, накопленный при предыдущих внедрениях. А вот отсутствие типового плана внедрения иногда приводит к интересным последствиям: я сам столкнулся с ситуацией, в которой внедренцы забыли, что «нарезка» 80 различных видов рабочих мест потребует определенного времени и определенных ресурсов.

> Многие западные системы поставляются с «готовой» методологией внедрения (фактически это перечень шагов развертывания системы в идеальном мире, некий «сферический конь в вакууме»). Частью таковой обычно является и типовой план, выглядящий примерно так:
>
> 1. Продажа лицензий и установка системы (конечно же, самый главный и своевременный этап);
> 2. Обучение;
> 3. Концептуальное понимание;
> 4. Функциональное описание;
> 5. Разработка;
> 6. Внедрение;
> 7. До свидания.
>
> Конечно же, хорошо, что типовой план существует, но про время на настройку восьмидесяти видов рабочих мест я бы уточнил дополнительно…
>
> Также в методологиях поставщика никак не предполагается то, что система будет внедряться на месте старой, что потребуется переносить данные, что данные эти в различных модулях не должны противоречить друг другу и что система должна как-то жить в переходный период… Ну, собственно, поэтому нас и нанимают… – *Д.К.*

«У нас свои средства описания проекта». На первый взгляд, ничего особо страшного. Только в этом случае вы сами уже не сможете понять, что же при обследовании поняли внедренцы, использовавшие эти средства. Еще хуже, когда это описание попало в ТЗ: когда вы при приемке системы начнете возмущаться, что система не делает половину того, что было обещано, а вторую половину делает, но не так, как это принято в вашей организации, вам покажут ТЗ и скажут, что сделано ровно то, что заказано.

Поэтому все функции системы в ТЗ должны быть описаны на русском языке. Хотят внедренцы сделать формализованное описание в терминах базового программного продукта – их право. Но при разрешении конфликтов определяющим должно быть словесное описание.

Готовность к созданию рабочей документации. Технический писатель. Сейчас для систем, которые продаются, какую-то эксплуатационную документацию худо-бедно делают. То есть документы, на титульном листе которых написано «Руководство пользователя» или «Руководство администратора», вам покажут. Только вам нужно сразу разобраться, не рекламные ли буклеты у вас в руках. Я однажды на рекламацию, что режим, описанный в руководстве, не работает, получил «тиражный» ответ разработчика: «Данная функциональность в системе предусмотрена, но еще не реализована».

Но все это верно именно для типовой, коробочной версии. Если систему дорабатывали по вашему заказу, то неплохо понять, есть ли надежда получить документацию, в точности соответствующую вашему релизу. Конечно, вы запишете это требование в договор или ТЗ, но будет неплохо, если вам еще покажут пальцем того сотрудника компании-внедренца, которому будет назначена роль технического писателя, а вы сможете убедиться, что этот человек в состоянии понятно писать на русском языке.

Я понимаю, что эффект от хорошего технического писателя у разработчика обычно смазывается отсутствием хорошего технического читателя у заказчика. Пусть так, но если сам разработчик поймет, что именно он сделал, то технический писатель свою миссию выполнил.

Кстати, помощь технического писателя в доводке системы бесценна. Мало того, что ему приходится пройтись по всему интерфейсу и выловить всех блох, пропущенных тестировщиками, он ведь еще выявляет все неописуемые элементы системы.

И вроде должно быть понятно: когда что-то работает так, что ни в сказке сказать, ни пером описать, это что-то надо переделать. Но объяснить это руководителю проекта, истекающему слюной по бонусу за закрытие этапа, удается чрезвычайно редко.

Личность демонстратора. Тестирование системы можно считать успешным, если человек, который показывает систему, вам совершенно несимпатичен, но система тем не менее вам нравится: *бойтесь харизматиков, системы приносящих.*

Как пытать разработчиков и внедренцев

Перечисленные ниже вопросы были сформулированы при поиске системы для торгового холдинга. Сформулированы давно, так что некоторые из них уже смотрятся диковато. Но менять я ничего не стал: все равно вы должны написать собственный список – я объект вашей автоматизации не изучал. Сделать это нужно заранее, до начала общения с продавцами программного обеспечения, так как каждый из них будет вешать вам на уши свою лапшу, а поскольку продают софт сейчас тоже профессионалы, очень может статься, что после квалифицированного общения вы «сами» решите, что перечень ваших требований совпадает с перечнем функций предлагаемой вам разработки. По той же причине посмотреть нужно существенно более одной системы.

Ответ на любой вопрос можно считать достоверным, если заявление представителей фирмы-разработчика подтверждается вашим тестированием системы или информацией независимых специалистов, которые эту систему эксплуатировали.

При общении с разработчиком нужно прежде всего определить уровень его понимания предметной области. Иногда у меня складывалось впечатление, что мы смотрим в одном направлении через очки с поляризованными в разных направлениях стеклами: там, где я видел стену, для них было чистое поле. Обращайте внимание на фразы типа «Ничего страшного, что в нашей системе товары считаются разными, если у них разные штриховые коды». Такие фразы показывают, что вместо подробного изучения реальной предметной области перед началом проектирования системы разработчики в основном ориентировались на придуманную ими модель, никак не связанную с реальной жизнью.

Если не очень понятно про штрихкоды.

Сигареты LM (красные) выпускаются по лицензии в сотне стран. Количество производителей в каждой стране может тоже измеряться десятками. Вся эта информация заложена в штрихкод (он же бар-код) стандарта EAN-13. Таким образом, количество различных штрихкодов для этого товара может доходить до нескольких тысяч. Но покупателя не волнует, какой штрихкод стоит на пачке сигарет. Как следствие, это не волнует ни мерчендайзера, который отвечает за наличие товара в зале, ни менеджеров, отвечающих за закупку этого товара и его доставку до магазина. Они все хотят видеть в информационной системе ровно одну строку, показывающую наличие товара в зале, на складе, в магазине и в пути. И цена продажи этого товара должна быть задана ровно одна. И если всех этих людей заставить

работать с сотней строк вместо одной, то можно услышать о себе очень много интересного.

Достаточно много сил и времени при изучении каждой новой системы у вас уйдет на овладение языком, который в ней и вокруг нее используется. Я не про язык программирования. Я про ту версию русского языка, которая использовалась разработчиками. Как я уже написал, словарь разработчика меняется гораздо быстрее, чем его коды.

> Однажды я даже получил официальное письмо на бланке с подписью и печатью, текст которого достоин опубликования:
>
> *«Доводим до вашего сведения, что с 01.01.2001 модуль Бухгалтерия называется Главная книга».*
>
> Без комментариев.

Ниже приводятся варианты толкования разработчиком некоторых терминов.

Логистика. Читаем в словаре: «Логистика – теория и практика управления материальными и информационными потоками в процессе товародвижения». Как дисциплина логистика занимается вопросами оптимизации при оформлении, перемещении и хранении товаров. В большинстве же систем модуль «Логистика» раньше назывался «Склад». Он считает складские остатки и ничего не оптимизирует. Но вам может встретиться и система, в которой модуль «Логистика» занимается учетом перевозок. И тоже ничего не оптимизирует.

Контрагент – лицо или учреждение, берущее на себя известные обязательства по договору. Это по словарю. Но в зависимости от системы справочником контрагентов будет называться либо справочник организаций, либо справочник (набор справочников), который используется в бухгалтерских и/или складских карточках и включающий как организации, так и физические лица, подразделения, темы и даже товары. Когда я попытался не так давно сам описать проект тиражируемой информационной системы, я обнаружил, что очень тяжело подобрать общее название для всех объектов и субъектов, фигурирующих, например, в договоре, накладной, платежном поручении (отправители, получатели, плательщики и т. п.) и в разрезе которых ведется материальный и аналитический бухгалтерский учет. Я бы назвал такие объекты фигурантами, но понимаю, что систему, где используется такая терминология, никто в России не купит.

> Что интересно, в отечественных системах часто один справочник – «контрагенты», а в большинстве зарубежных – и «поставщики», и «подрядчики, и „перевозчики“, и т. п. Как следствие, ответ на вопрос „«кто сколько нам должен?“«получается не совсем очевидным… – *Д. К.*

Бухгалтерский учет—термин используется как для метода учета, описанного еще Пачоли и использующего двойную запись, так и для части учета, предназначенной для демонстрации налоговой службе, в противоположность *управленческому учету*. Как разработчики используют последний термин, тоже нужно разбирать особо. Во всяком случае фраза «Управленческий учет построен по принципам бухгалтерского учета» оказалась для некоторых разработчиков не такой ясной, какой представлялось мне.

> А вот это исключительно наше изобретение. Во всех остальных странах управленческий учет – это не «черная касса», а просто более детализированный учет (в основном учет затрат и доходов). Управленческий учет в западном его понимании совсем не противоречит финансовому. – *Д. К.*

Кроме отдельных понятий оказывается смазанным также и смысл целых предложений. Высказывание «В системе поддерживаются стандарты бухгалтерского учета РСУ, IAS, GAAP» может означать:

1) что учет возможно организовать по любому из стандартов, но ровно по одному;

2) что, используя счета и проводки, сделанные по российским правилам, можно сваять отчет по IAS или какому-нибудь из GAAP;

3) что для одного из GAAP или для IAS используются свои счета и свои типовые хозяйственные операции.

> Нет такого стандарта, как GAAP. То есть совсем нет. Многие страны именуют свои правила учета «общепринятыми» (то есть generally accepted, или GAAP), поэтому аббревиатура бессмысленна без указания страны, откуда родом стандарт (например, US GAAP). – *Д. К.*

> Вот еще пример: пункт меню под названием «Распечатать жирорасчет по счетам-фактурам на заказ». До сих пор не знаю, что имелось в виду. Вообще,

перевод, выполненный лингвистами, не владеющими основными понятиями из предметной области, приводит иногда и к очень серьезным последствиям.

Пример из той же Axapta. В свое время термин invoice разработчик перевел (что характерно для западных систем) как «счет-фактура». Беда заключалась в том, что в те времена в законодательстве не существовало определенного срока, в течение которого нужно было выдавать эту фактуру, и были возможны ситуации, когда ее либо не существовало вовсе, либо она выдавалась со значительными задержками сразу на много поставок. В результате – либо в системе не следовало формировать фактуру до момента ее поступления (результатом чего становилась неучтенная задолженность), либо формировать (и получить косяк с формированием отчетности по НДС, в которой операция не считалась таковой без нужных бумажек). – *Д. К.*

Необходимо учитывать, что ответ на все задаваемые вопросы «ето могем» скорее характеризует безответственность отвечающего, чем качество системы. Старайтесь по каждому вопросу допытать разработчиков до чистосердечного признания, что эта возможность хотя и предусмотрена, но еще не реализована, а другая и не предусмотрена, но принципиально реализуема. Стоимость доработок, как показывает моя практика, может превзойти стоимость самой системы, а время их выполнения может быть больше времени разработки собственной системы.

Когда разработчик приводит примеры успешного использования, нужно выяснить, предлагается вам та же версия системы или один софт связан с другим только торговой маркой.

Учтите также, что, к сожалению, демонстраторы системы достаточно часто не разбираются в торговых технологиях и могут просто не понимать задаваемых им вопросов.

Даже ответам «Этим мы не занимались» и «Этого мы не предусматривали» нельзя слишком доверять: они могут характеризовать уровень знаний отвечающего, а не саму систему.

Но если вы услышите фразу: «Мы как раз сейчас разрабатываем версию системы, которая в точности удовлетворяет вашим потребностям», – улыбнитесь как можно лучезарнее и скажите: «Замечательно! Дайте мне знать, когда эта версия будет готова», – и бегите, бегите из того места, где вам это сказали!

Итак, вопросы.

1. Уровень интеграции (склад, магазин, бухгалтерия, финансовая деятельность, логистика, есть ли учет услуг, договоров, включающих и товары, и услуги, etc.), как связаны компоненты системы. Насколько хлопотно получить бухгалтерские проводки по документу, как привязать товарный документ к конкретным оплатам, как оплатить расходную накладную частично кассовым ордером, а частично возвратной накладной и можно ли это вообще сделать и т. д.

2. Идеология: от финансов, от документов, от товаров etc. (Разработчики обычно любят разглагольствовать на эту тему, однако смысл того, что они говорят, не всегда уловим. Иногда прояснить ситуацию позволяет простой вопрос: какой из модулей написан первым. Конечно, система «от склада» или «от бухгалтерии» звучит не так романтично и идеологично, зато дает определенное представление о том, как повернуты мозги разработчиков. Отрадно отметить, что сейчас на рынке появились системы, которые продумывались при проектировании целиком.)

> Еще любопытный момент: делалась система «от процессов» или «от структуры данных». Опыт подсказывает, что вторая система будет более жизнеспособной, так как при совершенно разных процессах данные, необходимые участникам, отличаются не так значительно.
>
> К сожалению, большинство зарубежных систем – процессоориентированные (то есть будут хорошо поддерживать разграничение ответственности при прохождении процессов, но доставят немало головной боли при попытке модифицировать процесс либо получить из системы нестандартный отчет). – *Д. К.*

3. Поддержка разработчиком изменений в законодательстве. (Не стесняйтесь спрашивать: «Что произойдет, когда Госкомстат заменит форму ТОРГ12 на ТОРГ13?» – в ответ иногда можно услышать и правду: «Сами исправите или нам денег заплатите».)

> И еще уровни прохождения заявки. То есть: сначала попугай внедренца, потом консультант внедренца, потом программист внедренца, затем (если ошибка глубоко) попугай поставщика системы, затем старший попугай поставщика системы, затем консультант поставщика системы… и так до двадцати человек, пока мы не узнаем, что наша ошибка исправлена не будет или будет исправлена в новой версии в следующем году. – *Д. К.*

4. Авторское сопровождение системы (нет, есть: бесплатно или платно). Его уровень (hot line, консультации у разработчика, консультации на месте, исправление обнаруженных ошибок, возможность модификаций, не связанных с ошибками, по запросу клиента)

5. Организация рабочего места (АРМ).

5.1. Набор функций жестко задан разработчиком (например, «АРМ бухгалтера/АРМ кладовщика/АРМ менеджера по продажам», причем эти АРМы организованы так, что менеджер по продажам не может посмотреть остатки склада) или произвольно конфигурируются администратором системы.

5.2. Варианты организации доступа к данным:

– можно ли разделить доступ к модулям, функциям, меню, процедурам, таблицам базы данных, полям таблиц, строкам таблиц, выделенным по каким-либо критериям;

– как выделяется: полный доступ/только чтение;

– для кого выделяется: описание доступа индивидуально для каждого пользователя, для групп пользователей, для типа рабочего места etc.).

6. Возможность одновременного ведения управленческого и официального бухгалтерского учета в необходимых стандартах на разных счетах. Только нужно убедиться именно в том, что все это будет работать параллельно, поскольку зачастую фраза «поддержка разных стандартов бухгалтерского учета» означает, что можно настроить ровно один из них.

7. Возможности защиты, охраны и обороны данных. Авторизация воздействий на систему. Специальные вопросы: насколько быстро ваша система будет вскрыта УБЭП; существует ли возможность подключения процедур «отбеливания» информации, в том числе в процессе штурма офиса «масками-шоу».

8. Работа корпорации (несколько юридических лиц и ПБОЮЛ, ведение раздельного и консолидированного баланса).

9. Возможность ведения баланса партнера (контрагента).

10. Работа с контрагентом-корпорацией.

11. Возможность одновременного ведения реального и официального учета по контрагенту (поставщик Вася может вполне официально работать под несколькими юридическими лицами, а вы должны уметь работать как с каждым юрлицом, так и Васей в целом).

12. Работа с несколькими территориями (филиалами, складами, магазинами, партнерами (контрагентами)). Обмен данными: методы репликации, уровень обобщения; технические

средства, возможные протоколы, интерфейсы и алгоритмы.

> Угу. Распределенка и что с ней будет, если грохнется канал. – *Д. К.*

13. Ведение учета по партиям товаров, возможные варианты определения партии (товары данной поставки, товары данного поставщика, товары данного поставщика с одинаковой ценой поставки, товары данного поставщика за определенный период времени etc.).

14. Подробность информации о контрагентах и производителях товаров. Для некоторых разработчиков факт, что производитель может быть и поставщиком, настолько удивителен, что «Поставщики» и «Производители» являются в системе никак не связанными сущностями, что, согласитесь, не слишком удобно.

15. Отслеживается ли товар в пути, деньги в пути. Как учитывать увеличение себестоимости товара в связи с его транспортировкой, хранением, обработкой?

16. Возможность вести учет в различных валютах, с несколькими обменными курсами, возможные моменты пересчета, ошибки округления и пересчета в документах и на карточках.

17. Различные упаковки и единицы измерения для одного товара (батоны, буханки и килограммы; бутылки и литры; далы (декалитры, приведенные к крепости напитков)). Возможности пересчета из одной единицы в другую (саморезы покупают килограммами, а продают штуками). Возможности добавления единиц измерения в процессе работы с товаром (хлеб захотели резать по полбуханки, жвачку продавать пластинками, а сигареты на штуки), получение данных и оформление документов в нужных единицах, ошибки округления при переводе из одной единицы в другую. Можно ли задать точность единицы измерения? (Был случай, когда у меня на складе осталось 0,1 бутылки коньяка.) Характеристики габаритов, объема, веса.

18. Открытость системы.

18.1. Возможность экспорта и импорта данных (текстовые файлы, MS Office, различные СУБД, бухгалтерские системы, программы «Клиент – банк», выдача информации в необходимых форматах для налоговой службы, пенсионного фонда, Госкомстата, службы занятости, служб алкогольконтроля etc.).

18.2. Возможность разработки собственных текстовых/табличных документов по данным, содержащимся в системе (генераторы отчетов и печатных форм).

18.3. Возможность подключения OLAP-продуктов или

встроенный OLAP. Если разработчик уверен, что у него «встроенный OLAP», то опять-таки придется разобраться, что он имеет в виду. Предполагаю, что вы такого и предположить не могли.

18.4. Передаются ли исходные тексты, есть ли возможность самостоятельной модификации структуры баз данных?

> А также исходные тексты чего и на каких условиях передаются. Например, я встречал продукты, где лицензировалось количество создаваемых программистом объектов. – *Д. К.*

19. Сетевые операционные системы, СУБД, операционные системы рабочих мест, метод работы с сервером (клиент-сервер, файл-сервер или их гибриды).

20. Ограничения по объемам баз данных и классификаторов, характеристики производительности

> Лучше, если на примере реальной инсталляции, так как такие ограничения могут быть неизвестны даже самим разработчикам. – *Д.К.*

21. Необходимые и допустимые технические средства. (Технические характеристики и возможные типы компьютеров, электронных весов, сканеров и принтеров штрихкодов, кассовых аппаратов, устройств работы с пластиковыми картами etc.).

22. Возможные классификаторы, уровень вложенности, возможности использования. Наличие размеров (одежда, обувь), цветов и других характеристик товаров. Работа с комплектами (наборами и сборными товарами), товарами в возвратной таре, учет тары, в которую укладывают сразу несколько видов товаров при отпуске со склада etc.

23. Способы калькуляции себестоимости, средневзвешенных цен, методики списания (по средневзвешенной, FIFO, LIFO, срок годности, минимальная/максимальная цена закупки etc.). Для корпоративных систем: возможность различной настройки для разных компаний корпорации.

24. Ведение прайс-листов (количество вариантов, валюты), возможные варианты скидок/наценок (оптовая, по общей стоимости закупки, по количеству единиц товара, по количеству наименований товара, по факту покупки конкретного товара, по времени покупки, по стоимости закупок данным клиентом за период времени, по категории клиента, индивидуальная для клиента etc.). Различные возможности получения вычисляемых

цен по базовым.

25. Комиссионная торговля, консигнация, реализация, экспедирование чужого товара, товарные кредиты, бартер, резервирование, автоматическое снятие резервирования etc. Оплата отдельных товаров из товарного документа. Фиксация цены закупки при оплате товара (в том числе и после его продажи).

26. Возможность контроля плановых запасов.

27. Автоматический учет пеней и штрафов (факт, прогноз и анализ).

28. Учет работы и оплаты менеджеров, торговых агентов, коммивояжеров и др.

29. Слияние/разделение товаров (один товар зарегистрирован в системе с разными кодами, разные товары попали в систему под одним кодом).

30. Слияние/разделение контрагентов (ИНН попадает в систему позже ввода в систему контрагента).

31. Наличие утилит для обрезания и сжатия разбухшей базы данных.

32. Наличие интегрированных в систему средств архивирования данных.

33. И, если честно, все это работает?

* * *

Итак, вы получили ответы на свои вопросы. Что дальше?

Дальше вы, как и предполагалось с самого начала, обнаруживаете, что полностью вам не подходит ни одна из систем.

Теперь сразу же и решительно вычеркните из списка системы, которые точно не годятся, как бы вам ни нравились их интерфейсы, заложенные в них идеи и технологии. Постарайтесь проявить твердость и в общении с начальством по этому поводу. Нужно жестко отсечь системы, использующие СУБД, которые:

– не поддерживают объемы данных, как минимум в десять раз превосходящие объемы, заложенные в проект;

– не поддерживают одновременную работу числа пользователей, превышающего проектное в два-три раза;

– не поддерживают работу в сетях соответствующего масштаба или репликации в распределенной базе, и многое другое, что не лечится, если не было заложено в программное ядро продукта в самом начале его разработки.

Все-таки и интерфейс имеет значение. В системе, в которой при вводе заказа оператор должен при создании каждой новой строки кликнуть на пять кнопок (мышкой) и вручную каждый раз перебрать весь список товаров, точно не подойдет для организации, в которой продавец должен по телефонному звонку набирать заказы на двести строк.

Нужно только определить, для каких рабочих мест интерфейс является критичным, а для каких – нет.

Но не стоит путать удобный интерфейс и графический а-ля Windows – в упомянутом выше случае работа мышкой не всегда лучший вариант. – *Д. К.*

Вам будет тяжело избежать соблазна самому и уберечь от него своих шефов: системы, которые вы отвергнете, будут иметь самый красивый и удобный интерфейс и как нельзя лучше подходить для решения двух-трех частных задач, о которых именно сейчас думает начальник.

Особо остановлюсь на тестировании объемных и временных характеристик. В этом вопросе не следует доверять ни заверениям разработчиков, ни документации на СУБД. Вы должны проверить это сами. Понятно, что создать тестовую базу в три гигабайта у вас вряд ли получится. Для проверки работоспособности на таких объемах информации вас должны свести с организацией, которая уже использует соответствующую систему и оперирует соответствующими объемами информации. А вот как система работает с документом в тысячи строк (нормальный размер акта инвентаризации магазина), вы должны проверить сами.

Экстраполяции временных характеристик вам не помогут: они могут изменяться нелинейно. И если даже сохранение документа в десять тысяч товарных позиций происходит в сто раз медленнее, чем у документа в сто товарных позиций, то скроллинг строк этого «монстра» может происходить и в десять тысяч раз медленнее.

На отобранные системы примериваются задачи вашего проекта, отмечаются все места, «где видно голое тело» и «где скоро порвется», и подбирается набор заплаток (организационных и программных), которые можно в этих случаях применить.

К наличию у системы возможностей, которые вам не нужны, следует относиться осторожно. С одной стороны, они, конечно, могут понадобиться в дальнейшем, но с другой, они обычно приводят к ухудшению потребительских свойств частей, которые вам нужны. Система становится менее удобной, так как

усложняется ее интерфейс (лишние вопросы, более длинные меню, необходимость вводить данные, которые не используются), и более объемной и медленной.

По итогам анализа систем вы готовите соответствующий документ, включающий также и стоимость реализации проекта с учетом использования каждой из систем-кандидатов, и начальство делает свой выбор.

Повторяю. Выбор основного программного продукта для автоматизированной системы управления масштаба предприятия (корпорации) делает *первый руководитель или высший руководящий орган предприятия (корпорации)* по вашему представлению. Значение этого шага гораздо выше, чем просто покупки программы (пусть даже очень дорогой). Реально оказывается, что принятое решение может на годы определить возможности и направления развития фирмы, и пусть руководство об этом знает.

Выбор системы руководством, конечно же, не спасет вас в случае неудачи от виселицы. Но вы пойдете на нее с чистой совестью.

Правда, руководство само тоже не всегда склонно принимать решение и произносит сакраментальную фразу: «Давайте проведем тендер».

Красивое слово «тендер»

Слово это очень любят проверяющие органы и руководство уровня, на котором информация добывается нажиманием на подчиненных, а не на кнопки клавиатуры. Выбор проектного решения на основе конкурса – дело, конечно, соблазнительное, но какие у вас есть варианты?

1. Каждая из компаний, согласная участвовать в тендере на поставку и внедрение информационной системы, проводит самостоятельное обследование вашей организации и предлагает свой вариант решения ваших задач с указанием стоимости решения. Обычно обследование организации одной компанией занимает не менее месяца. Сколько компаний выдержат обследуемые сотрудники вашей организации и сколько времени все это продлится?

2. Вы сами составляете подробный документ с описанием информационных потребностей и рассылаете его по участникам конкурса. Сроки тендера в таком случае сокращаются, а сотрудники вашей организации остаются живы. Но вот что поняли участники тендера из вашего документа, станет ясно

только в момент, когда уже установленная система не сможет выполнять возложенные на нее функции. Да и многие ли компании, искренне желающие внедрить у вас систему и объективно оценить затраты на нее, согласятся на такой вариант тендера?

> Как третий вариант, можно привлечь стороннюю компанию («консультант по работе с консультантами»), которая проводит обследование и составляет описание требований. Однако при этом желательно, чтобы такая компания участвовала во внедрении и дальше, беря на себя часть работы по постановке задачи и контролю ее реализации.
>
> В любом случае, когда требования формулируются сторонним консультантом, необходимо, чтобы в компании были люди, которые понимают и согласны с тем, что написал консультант (иначе после его ухода компания окажется в ситуации, когда нужно строить коммунизм, имея на руках только «Капитал» Маркса).
>
> И даже привлечение стороннего консультанта не спасает от того, что написано выше: решение будет принимать руководство компании. – *Д. К.*

При любом варианте проведения тендера с его помощью можно определить только наиболее дешевое из решений, которые сочли приемлемыми сами участники конкурса. А что делать с такими понятиями, как адекватность, удобство, полнота, которые количественно оценены быть не могут?

Мне кажется, что выбор информационной системы на основе тендера с формальным алгоритмом подведения итогов (стоимость, срок внедрения, даже балльные экспертные оценки, перемешанные в экзотической весовой формуле), очень напоминает игру в «русскую рулетку», когда в барабане револьвера нет пустых гнезд.

> Есть некоторые решения, которые, возможно, являются оптимальными, но не могут быть выявлены при помощи тендера. Это, например, внедрение вместо одной системы комплекса специализированных систем (кому-то придется собирать нескольких поставщиков в одну команду, способную вместе выступить на тендере) или заказная разработка всей или части системы. То есть перед проведением тендера нужно быть на сто процентов уверенным, что выбор имеет смысл только среди тех поставщиков, которые приглашены в нем участвовать. –

Д. К.

Если же по материалам, присланным на тендер, решение принимаете вы и ваше руководство, то… я это уже описал в предыдущих главах.

Подбор персонала

Создание команды, способной внедрить систему, – это искусство, а не ремесло. Можно дать определенные рекомендации, можно с большой долей уверенности сказать, когда команда работать не будет, но я не могу обещать, что, выполнив все рекомендации, вы получите работоспособный коллектив. Даже призыв «Делай как я» и подробное описание способов, которыми пользовался я, чтобы заставить эффективно работать своих и чужих сотрудников, здесь не помогают. Прежде всего потому, что этот коллектив начинается с вас и является вашим продолжением.

То, что вы должны подобрать, – это не бригада программистов, хотя в каком-то виде она в вашу команду, по всей видимости, войдет, но не с нее начинается и не ею заканчивается.

Группа, занятая разработкой и внедрением, состоит из трех частей: собственно специалисты отдела автоматизации, прикомандированные сотрудники других подразделений вашей фирмы и сотрудники, ведущие ваш проект со стороны разработчика (когда есть внешний разработчик).

Пользоваться прикомандированными сотрудниками иногда может оказаться удобнее, чем перетаскивать их со скандалом к себе в отдел: умного вам никто не отдаст, а дурак в отделе автоматизации обладает гигантской разрушительной силой.

Сейчас любая относительно крупная организация уже имеет какие-то элементы автоматизированного учета (есть хотя бы компьютер, какая-нибудь бухгалтерская программа; если как-то автоматизирован складской учет, то обычно есть и локальная сеть). Персонал, который все это обслуживает, почему-то обычно называют мальчиками или программистами, хотя, как правило, эти люди программируют разве что bat-файлы. Американцы называют таких сотрудников IT-person, не отличая тех, кто больше нажимает на кнопки, от тех, кто больше крутит отверткой. В последнее десятилетие и в российских ИТ-кругах для таких людей появилось название «эникейщик». Квалификация этих специалистов слабо связана с зарплатой и варьируется от уровня advanced lamer до вполне серьезного.

В более крупных фирмах разделяются функции системного инженера (специалиста по железу и операционным системам), системного (сетевого) администратора, администратора баз данных и, может быть, программиста для доводки и текущих модификаций программного обеспечения. Эти функции в разных сочетаниях распределяются по имеющимся людям.

Для создания и внедрения новой автоматизированной системы или ее частей потребуются и новые функции, хотя до определенных пределов их можно возлагать и на тех же сотрудников.

Во-первых, вам потребуются **квалифицированные постановщики**. Первый и главный из них – это вы сами. Однако будь вы хоть семи пядей во лбу, вряд ли вам удастся описать все задачи вплоть до уровня, доступного программисту-кодировщику или внешнему разработчику (во втором случае подробность проработки должна быть еще выше, чем в первом). Кроме того, как я уже говорил, для получения хорошего результата постановщик должен знать предмет лучше сотрудников, работающих в соответствующем направлении, что практически невозможно для одного человека. Я, например, вполне прилично разбираюсь в технологиях склада, магазина и оптовых продаж, то есть свободно владею менеджерским, складским, программистским и русским языками. На этих языках я понимаю их носителей, могу сформулировать свои мысли и, самое главное, думать сам. А вот бухгалтерский язык для меня – иностранный. Я понимаю написанное на нем другими, если напрягусь, и сам могу написать шаблоны проводок для хозяйственной операции, но у меня не могут появиться *идеи* на этом языке.

И я счастлив, что со мной много раз работали люди, мозги которых повернуты на некоторый угол по отношению к моим: понимая и складские, и торговые технологии, и даже что такое алгоритм, они умеют мыслить по-бухгалтерски.

Мало их… Я за свою жизнь встретил только трех, и все – женщины: две – великолепные бухгалтеры, одна – исполнительный директор, которая за своих бухгалтеров думает. Знаю я и мужчин, отлично ориентирующихся в бухучете, но они все почему-то занимаются не самим учетом, а его автоматизацией. Впрочем, «за свою жизнь» – это слишком сильно. Просто лет восемь назад я сам в бухучете ничего не понимал, а чтобы осознать величие бухгалтера, надо в его деле хоть что-то смыслить. Теперь я смыслю, но именно что-то. Креатив в этой области мне по-прежнему недоступен.

«Нам не нужна отдельная система контроля исполнения поручений, все можно сделать в модуле „Бухгалтерия“ нашей системы: когда поручение дается, датой отчета по поручению делается проводка на штраф ответственному, а если он поручение случайно выполнит, то проводка сторнируется».

Без комментариев.

Квалифицированные постановщики ныне редкость. Как ни странно, тяжелее всего найти людей, владеющих русским языком, что в результате приводит и к косноязычию на уровне программных кодов.

Быль 1. В 1995 году подобрал я компанию, готовую поставить информационную систему для фирмы, в которой я тогда работал. И стали мы обсуждать, как эту систему настроить. И мне впервые назначили менеджером проекта человека, для которого правильнописание было тайной за семью печатями.

Через некоторое время я сообщил их директору, что долго такого не выдержу, и попросил заменить внедренца или хотя бы обучить его пользоваться проверкой орфографии в Word. Следующий документ он мне вручил, торжественно заверив, что теперь, получив взбучку, он проверяет тексты очень внимательно. Документ назывался «МАТЕРЬЯЛЫ К ЭСКИЗНОМУ ПРОЭКТУ». В параметрах Word было указано не проверять слова из прописных букв…

Быль 2. Очередная организация наняла меня слишком поздно: сторонние внедренцы уже сделали вид, что провели предпроектное обследование. В результате еще месяц ушел на переделку технических заданий на доработки. Одним из требований, на котором я настаивал, было добавление в справочник контрагентов дополнительного поля для аббревиатуры в пять символов. Сокращенное название контрагента в выбранной тиражируемой системе занимало двадцать символов и не влезало в шахматки, которые широко использовались в организации.

Требование для этой системы пустяковое, руководитель проекта от внедренцев на него сразу по телефону согласился, но в очередной версии ТЗ аббревиатуры не оказалось. Звоню, удивляюсь, в ответ слышу: «Ах, простите, щас добавим». Приходит следующая версия ТЗ – нужного пункта нет. После трех итераций еду к разработчикам и понимаю, что единственной проблемой для включения требования в ТЗ было неумение написать слово «аббревиатура» менее чем с двумя ошибками кем-либо из внедренцев. А Word больше одной ошибки не исправляет…

Бригада программистов обычно бывает нужна, при этом

роль **главного программиста** можете играть и вы, если вам удастся разумно ограничивать себя на этом поприще.

Соотношение между численностью своих программистов и программистов разработчика, занимающихся вашими задачами, зависит от наличия, качества и стоимости сопровождения, а также от ваших возможностей и желания сопровождать систему самостоятельно. Зачастую вам было бы и дешевле, и удобнее программировать какие-то куски системы самим, но разработчик отказывается их поддерживать в следующих релизах системы.

Два типа программистов и особенно их соединение в одном лице категорически противопоказаны при разработке и сопровождении любых систем – это *интерпретаторы* и *исследователи*.

Интерпретаторы

Они люди по-своему талантливые, хотя часто еще и абсолютно невежественные. Главной их особенностью является бурное воображение, не позволяющее адекватно воспринимать окружающий мир. В голове у таких полная картинка возникает сразу после поступления первых битов информации. Дальше попыток согласовать появившуюся картинку с информацией, продолжающей поступать извне, не происходит. Их заменяет придумывание предметной области и придумывание пользователей системы вместе с их потребностями.

Интерпретаторы практически никогда не понимают, но и не уточняют поставленную задачу, поскольку заранее знают, что вам нужно, гораздо лучше вас. Письменная формулировка задания дела не спасает, поскольку они не дочитывают написанного. Следующий диалог отражает суть проблемы и практически не утрирован:

– Дима, закрась, пожалуйста, этот рисунок красным. Это срочно.

– Я уже сделал.

– Дима, а почему все зеленое, я же просил красным?

– Но ты же сказал «закрась», а с буквы «з» начинается именно зеленый…

Наиболее печальный результат получается при попытке использовать интерпретатора для сопровождения существующий системы. В самых тяжелых случаях его не удается заставить прочесть и понять чужой программный код. Мне пришлось некоторое время просуществовать в одном подразделении с индивидом, считавшим себя крутым программистом и искренне уверенным, что чужой программный код вообще невозможно изучить. Разуверить его у меня не получилось, но получилось

уволить.

Интерпретаторов более раннего возраста удается убедить посмотреть чужие коды, но эффект от этого не совпадает с ожидаемым: «понимание» функционала, реализованного изучаемым текстом, приходит к ним после прочтения первых трех строк. После этого, если не удается принять специальных мер, код правится и внедряется в программное обеспечение действующей системы. Система сразу начинает плясать краковяк, а на вопрос «Что произошло?» следует ответ: «Оказывается, эта процедура кроме того, что я думал, делает то-то и то-то. Но мне об этом никто не сказал».

Исследователи

Эти постоянно хотят осваивать что-то новое, что само по себе совсем не плохо. Но, в отличие от адекватных программистов, немедленно применяют изученное при решении текущей задачи. Собственно, выражение «колоть орехи микроскопом» уже отпраздновало свое столетие. Но если раньше от таких действий страдал только микроскоп, а орехи еще можно было съесть, то после аналогичных воздействий программиста еды может не получиться совсем. То есть по эффекту получается уже гораздо ближе к «стрелять из пушки по воробьям». Таким приходит в голову прикрутить к разрабатываемому электронному калькулятору оракловую базу данных для хранения промежуточных результатов вычислений, хотя в основном проекте используется MS SQL-сервер, который был интересен исследователю в прошлый раз, пару месяцев назад.

Из достижений исследователей за последние годы больше всего порадовали запись настроек и протоколов прикладной программы в реестр Windows и хранение информации внутри базы данных MS SQL в формате xml.

Для эксплуатации некоторых систем (пальцами не показываю) необходима новая разновидность сотрудников: **бухгалтерские программисты**. В идеале это люди, знающие и понимающие бухучет (зачастую в отличие от бухгалтеров организации) и умеющие записывать алгоритмы на специальном языке, который разработчики почему-то называют параметрической настройкой.

Кроме того, потребуется **технический персонал** для закупки и настройки компьютеров, генерации общесистемного софта, организации локальных сетей и интранета, кодировщики для работы в генераторах отчетов и печатных форм и программирования простейших отчетов и функций и сотрудники

для работы в качестве дополнительных операторов и обучения штатного персонала при запуске системы или ее новых функций.

Для выполнения всех функций последнего абзаца я обычно набираю «на временную работу» однородный штат ребят со средним техническим образованием или студентов последних курсов – молодых, холостых и относительно дешевых, без комплекса старухи из «Сказки о рыбаке и рыбке». Они последовательно выполняют все задачи проекта, но до следующего проекта добираются немногие: большая часть остается по дороге сисадминами, администраторами баз данных и начальниками службы подготовки данных. Меня это как человека радует, но персонал приходится снова добирать и обучать. Только осторожней с *кнопкодавами*.

Кнопкодавы

Этот тип сотрудников мне в прошлом веке практически не встречался. Может быть, просто везло, или это следствие нашего приближения к Западу с помощью всеобщего отупения. Сейчас им несть числа. Поэтому, когда неизвестный мне «системный инженер», получив информацию, что какое-то приложение работает не так, как следует, предлагает мне прежде всего переустановить операционную систему, я начинаю смотреть на него с опаской.[3]

Дураки, конечно, и раньше встречались. Они всегда были, есть и будут. И непрофессионалов всегда было сколько угодно: «Сотрудники уносили перфоленты домой, чтобы использовать их в туалете» – это известный в 1970-х писатель-фантаст, который, наверное, сам дуршлаг использовал для тех же целей; «Я напечатал вопрос на пишущей машинке и прочел ответ на перфоленте» – это журналист образца 1976 года.

Но сейчас появилось множество людей, работающих в области вычислительной техники и информационных технологий профессионально (в смысле за деньги) и не понимающих в этой области ничего. То есть абсолютно ничего.

Как они появляются, приблизительно понятно. Вера в чудеса неистребима, хотя сейчас вряд ли кто-то допускает мысль о существовании гаррипоттерных волшебников с седой бородой и волшебной палочкой. Для многих главным источником чудес стал компьютер, а главным волшебником – человек, который умеет с ним общаться. Желание тоже стать волшебником подсказывает самый доступный способ его осуществления: выучить заклинания.

Заклинаний они действительно знают много, гораздо больше меня, и пальцами шевелят с поразительной быстротой.

Полное незнание английского помехой не является: заклинаниям положено быть непонятными. То, что сейчас виндоусы в большинстве случаев русифицированы, тоже почти не помогает: они не понимают смысла и по-русски. Главное – в правильной последовательности надавить кнопки. Объяснить, что они делают и, самое главное, зачем, они не могут: для этого нужно хоть немного понимать, как функционирует компьютер, что такое процесс операционной системы и т. д.

Заклинания бывают специальные (приворот, вызов демонов… простите, подключение к принтеру или к локальной сети, установка скринсейвера или машущего хвостика вместо курсора мыши) и общие, на все случаи жизни (простое: Ctrl + Alt + Del, крутое: переустановка виндоуса).

К общим заклинаниям относятся Заклинание Счастья (установка найденного в Интернете дефрагментатора памяти под операционной системой, которая имеет свой дефрагментатор), и разгон процессора. Компьютер начинает глючить, виснуть. Истинный кнопкодав при этом причинно-следственной связи со своими же действиями не замечает и меняет материнскую плату. Или повторяет заклинание.

Попытки выяснить, для каких задач ему не хватило штатной тактовой частоты, для каких целей установлен дефрагментатор, приводят лишь к недоуменному взгляду: вопрос непонятен.

Для заклинаний существует общее правило: если заклинание не помогло, то его надо повторить. Я лично наблюдал пять переустановок Windows XP на одном из компьютеров, у которого перед этим был разогнан процессор.

Появление новых версий любых программ вызывает у кнопкодава неустранимый зуд от желания ее немедленно установить. Зуд усиливается в тех случаях, когда установка новой версии уничтожит всю информацию, накопленную с помощью старой. Удовлетворение наступает лишь в случае установки новой неотлаженной версии на всех доступных компьютерах. Опять-таки совершенно бесполезно спрашивать, какие дополнительные функции в новой версии почтовой программы были настолько необходимы хотя бы ему, что он мне эту версию установил, загубив мою переписку за два года. В ответ вы получите лишь искреннее непонимание, зачем шефу нужны старые письма…

«Когда у человека есть только молоток, весь мир кажется ему гвоздем». Абрахам Маслоу, кажется.[4]

Сэмпл-программеры

Дальнейшее развитие кнопкодавов. Люди, изучившие какой-

то язык программирования, но не умеющие составлять алгоритмы. Что существуют инвалиды, лишенные соответствующей способности, я обнаружил, еще преподавая в институте. Но в те давние времена, даже получив вузовский диплом соответствующего профиля, они старались программ не писать. Теперь пишут, благо языковые средства сейчас алгоритм успешно маскируют. Делается это с помощью образца программы, выполняющей похожие действия (отсюда и название подвида: sample-programmer). В такую готовую программу добавляются заклинания с указанием необходимых полей базы данных или переменных. Если не получилось, то добавляются другие заклинания. Как ни странно, результат иногда достигается и таким способом. Только взглянув на текст, вы неожиданно обнаруживаете, к примеру, что из базы данных информацию получали тремя последовательными запросами, после чего были использованы только результаты второго.

Шаловливые Ручки (ШР)

В отличие от предыдущих подвидов этот известен мне с молодости. В 70-х такой персонаж выдергивал перфокарту из середины ненадписанной колоды перфокарт полуметровой толщины, «чтобы посмотреть качество работы выходного перфоратора», и зачищал надфилем посеребренные контакты. Со временем действия менялись, и к концу 90-х ШР новой генерации уже форматировал заполненный информацией жесткий диск на компьютере директора, переданном для апгрейда.

Парадокс состоит в том, что ШР не является кнопкодавом. Вне состояния умопомрачения он прилично знает свою специальность и в состоянии рассказать и как функционируют компьютер и система, и что нужно сделать для правильного решения поставленной перед ним задачи.

Конечно, таких людей специально никто на работу не берет, но со временем ШР заводится почти в каждом отделе соответствующего профиля.

Еще быль. Шаловливые Ручки в очередной раз уложил новеловский сервер. Когда сотрудники отдела в мыле возвращаются на свои места, перезапустив сервер, бежать до которого нужно было около полукилометра, и восстановив работоспособность всех крутящихся под ним задач и систем, я в сердцах говорю:

– Ну что мне с тобой делать?

Остальные сотрудники наперебой начинают предлагать, *что* с ним нужно сделать.

– И ведь это не лечится… Хотя… Есть у Босха такая картина –

«Операция на глупости»…

После пятиминутной паузы ШР робко спрашивает:

– А Босх – это из PowerPoint?

HR – звук, издаваемый во сне

Совершенно не понимаю, чем термин «подбор персонала» хуже, чем «human resources». Наверное, импортные слова всегда звучат красивше, поэтому люди, которые этим делом занимаются, выбрали для самоназвания именно второе, после чего немедленно раздулись от гордости. Лучшие представители этой профессии действительно в состоянии отобрать резюме кандидатов из доступных баз данных по ключевым словам и после собеседования дать толковую психологическую характеристику интервьюируемого. Но и они обычно не разбираются в ИТ и, если даже понимают, что программисты чем-то отличаются от сисадминов, совершенно не в состоянии определить квалификацию ни тех ни других. Так что это тоже будете делать вы.

В одной из компаний я отвечал и за набор операторов, которые круглосуточно вводили информацию по товародвижению в систему. Операторы работали в помещении склада, который, в свою очередь, располагался на территории железнодорожной станции. Каждый раз, объяснив кандидату, как добраться до места, я очень переживал, что моих объяснений недостаточно и бедняга обязательно заблудится. А потом до меня дошло, что если этот бедняга заблудится после моих объяснений, то это не очень страшно, во всяком случае для меня и для фирмы: ведь он точно так же не поймет и моих объяснений его обязанностей, правил и технологии работы и т. д. И очень может быть, что это не вина кандидата, может быть, это я не умею понятно объяснять, но в любом случае толка от нашей совместной работы будет мало. Значит, мне нужен другой оператор. Так умение разыскать офис, а в нем меня стало первым отборочным испытанием. Вторым было заполнение анкеты: если я не мог прочесть, что в ней написано, кандидату отказывали. Я отдавал себе отчет, что таким образом отвергаю гениев. Но если нужны операторы, то гениев набирать не стоит.

Для сотрудников более высокого уровня первым тестом является резюме. Если оно составлено с существенными нарушениями существующих традиций, то причин этому могло

быть много:

– человек не слишком заинтересован в работе и поэтому не выяснил, как составляют резюме (нет достаточной мотивации);

– человек выяснял, как составлять резюме, но не смог этого понять (нет способности обучаться);

– человек понял, как составлять резюме, но не смог его составить (недостаточные интеллектуальные способности);

– человек знал, как составлять резюме, мог его составить, но не стал этого делать (косит под гения).

Но в любом из перечисленных случаев кандидат не нужен мне в качестве сотрудника.

Основное время собеседования я посвящаю ровно двум вопросам:

1. Расскажите, чем вы занимались на предыдущем месте работы;

2. Расскажите о проекте, который вам больше всего запомнился.

Вас не должно смущать, что на предыдущем месте кандидат занимался совсем не тем, чем должен заниматься у вас. Я все больше убеждаюсь, что человек, понимающий, что он делал раньше, и способный об этом рассказать сейчас, имеет больше шансов понять свои нынешние обязанности, даже если у него таких не было. Конечно, из того, что кандидат был хорошим художником, не следует, что он станет у вас хорошим виолончелистом. Но если кандидат говорит, что писал программы для рабочего места менеджера издательства, но не может ответить на вопрос, чем же занимается менеджер издательства, потом добавляет, что разрабатывал сайт, но впадает в ступор от вопроса, про что был сайт, пожелайте ему успехов в поисках работы, но не у вас.

И если кандидат на должность администратора информационной системы говорит, что у него были в системе два территориально разделенных склада, между которыми перемещали товар, но не помнит, каким способом достигалось соответствие расходной накладной склада-источника приходной накладной склада-приемника, то про него тоже многое становится ясным.

Еще одно общее наблюдение. Если в процессе решения предложенных задач кандидат принимает позу роденовского «Мыслителя», брать его на работу не следует: он не умеет думать, он только изображает процесс мышления. Знаете ли вы, что Роден эту скульптуру в первом варианте поместил в композицию «Врата ада»? И был абсолютно прав: человек автоматически принимает такую позу в те моменты, когда думать уже поздно. Уж какой

умник впоследствии убедил Родена, что это «мыслитель», я не знаю. Но если вы случайно встречали в жизни людей задумавшихся, то, наверное, заметили, что они застывают ровно в том положении, в котором их застала мысль: кто с открытым ртом, кто с приподнятой рукой. Глаза при этом зафиксированы на произвольном предмете, желательно неподвижном, но предмета этого не видят. А «приняв позу», думать не получается: вся энергия уходит на поддержание позы.

Отбор программистов

Отдельную главку хотелось бы посвятить отбору программистов в связи с особой важностью этого процесса.

Квалифицированные ИТ-специалисты сейчас вообще в дефиците, а поиски программистов в непрофильную фирму становятся занятием почти безнадежным. При практически одинаковом уровне зарплат работа в компании – разработчике софта или у ИТ-консалтера психологически намного приятнее: гораздо ближе к нетленке и дополнительным знаниям при гораздо меньшей ответственности.

Одно дело – прочесть о своей ошибке на экране системы сопровождения проектов, и совсем другое – наблюдать, как весь офис не встает из-за компов до часу ночи, потому что ты поставил конец цикла на строчку ниже, чем нужно, даже если тебе никто ничего по этому поводу не говорит. А бывает, что и говорят, много и цветисто.

Один из моих сотрудников, которого удалось переманить у фирмы, поставившей нам информационную систему, через полгода сбежал обратно, хотя мы холили его и лелеяли, да и зарплату были готовы платить в 2,5 раза больше той, на которую он вернулся.

Особенно приятно программировать в организациях с круглосуточным режимом работы, когда тебе звонят в четыре утра и сообщают, что в накладных перестала подставляться цена, а их уже нужно печатать, чтобы развезти товар по магазинам.

В итоге к такой работе стремятся в основном только желающие быть «самым крутым программером ликероводочного завода», но именно такие не слишком нужны вам. Вам нужны хорошие программисты.

Хорошие программисты встречаются редко, поэтому им много что прощается. Но я бы хотел предостеречь молодых читателей от стандартной логической ошибки: хороший программист – это не тот, кто опаздывает на работу и на любую

встречу, в какое бы время она ни происходила, не тот, кто не меняет носки и не принимает душ, и не тот, у которого последствия насморка свисают на небритый подбородок. Хороший программист – это тот, кто адекватно понимает задачи и в состоянии их решить с помощью программных средств быстро и без ошибок.

Поверьте, среди хороших программистов встречаются и аккуратные, и чистоплотные. Я встречал даже таких, которые свою кофейную кружку моют каждый день.

Поэтому, чтобы стать хорошим программистом, не нужно отпускать бороду, переставать мыться и регулярно портить воздух в помещении, а нужно:

1) учиться понимать, что в отличие от языка программирования внешний мир нельзя узнать и описать полностью;

2) учиться понимать задачи, даже когда их формулируют на неформальном языке, даже когда их формулируют не совсем корректно: внешний мир сам не всегда корректен;

3) учиться придумывать и описывать алгоритмы, желательно оптимальные;

4) учиться работать в команде;

5) учиться работать внутри ПО больших эксплуатируемых систем, не напоминая слона в посудной лавке.

Отбор программистов для работы всегда напоминает лотерею. Что и как ни проверяй, понять, в состоянии ли работать программист в вашей команде, удастся в лучшем случае через полгода. Зато тех, кто точно не будет в состоянии работать, можно отсеять на стадии собеседования.

В резюме программистов я сразу же смотрю раздел специальных знаний. И если в качестве языков программирования перечислены Word, Excel и html, резюме отправляется в корзину незамедлительно.

Просьба принести на собеседование несколько страниц исходных текстов своих программ на любом носителе почему-то сразу отсеивает процентов десять кандидатов. Ясно, что у вас нет никакой возможности проверить, написан ли исходник самим кандидатом, но вы можете понять, какой текст программы кандидат считает годным для предъявления новому работодателю, то есть качественным.

Кстати, совершенно не обязательно, чтобы кандидат в программисты очаровал вас во время интервью. Программист – не сейл, поэтому может быть и стеснительным, даже слегка аутичным, и свои успехи совершенно не обязан уметь

рекламировать. Более того, к тем, кто себя умеет хорошо продать, следует относиться настороженно: совмещение профессий (а особенно, таких как сейл и программист) встречается редко.

Конечно, на интервью надо поверять уровень знания кандидатом программных средств, которые он продекларировал как использовавшиеся. Даже если работать придется с другими языками программирования и СУБД. Любой квалифицированный программист хоть какие-то языковые средства должен знать хорошо. Людей, поклоняющихся языку программирования как божеству, на работу стараюсь не брать. Кандидатов, которые начинают петь песни «Си рулит, Паскаль отстой», «Слава Юниксу» и т. п., даже когда их не просят петь вовсе, я отсеиваю. Они обычно не понимают, что язык – всего лишь рабочий инструмент, а не волшебная палочка. Я до сих пор убежден, что человек, который в состоянии формулировать алгоритмы на одном языке программирования, достаточно быстро и без проблем освоит любой язык. Но вот отсутствие алгоритмического мышления не может заменить даже знание всех языков мира.

> Одну задачу я даю практически всем. Вот эту.
>
> *Написать программу на любом языке программирования.*
>
> *Входные данные в формате чч: мм:*
>
> *1) время прибытия поезда по расписанию; 2) время фактического прибытия поезда на станцию.*
>
> *Выход: текст: «Поезд опоздал» / «Поезд пришел раньше» / «Поезд пришел вовремя», время задержки или опережения.*
>
> Поскольку у читателя совершенно случайно может возникнуть желание решить приведенную задачу, разбор ее решения я поместил в Приложение 1.

Конечно, посмотреть, как кандидат решает задачи, которые вы даете, интересно и познавательно. И некоторую информацию вам это даст. Однако не забывайте, что **результат любых тестов объективно оценивает только способность испытуемого проходить тесты соответствующего вида**. Вам нужен стайер, который должен бегать кроссы по пересеченной местности в составе команды, а вы пытаетесь его оценить по тому, как он бегает спринтерские дистанции в одиночку.

Подряд

Часть работ может быть возложена на внешних подрядчиков.

Основным критерием для возможности передачи работ служит одновременное выполнение перечисленных ниже требований:

– возможность четко сформулировать задачу;

– возможность выполнить эту задачу независимо от остальных задач;

– возможность быстро проверить результат и принять работу.

Таким образом, вполне годятся для передачи в подряд организация локальной сети, написание отчета по заранее описанной базе данных, но не вставка куска в чужой код. Не забывайте, что оговоренные с подрядчиком сроки должны быть как минимум в два раза меньше тех, которые вам реально необходимы. Впрочем, это не сильно помогает.

Стиль руководства

В математике функции бывают сразу четными и нечетными. Только это все подфункции тождественного нуля. А начальники умеют быть посредственными и непосредственными одновременно.

Про то, как нужно руководить людьми, написана куча литературы, особенно в Соединенных Штатах. Часть из нее даже полезно прочесть, хотя это и не может заменить вашего личного опыта и мозгов.

Сам я вряд ли могу научить, как надо руководить, зато точно могу сказать, как не надо.

Приводимый ниже текст написан мной несколько лет назад. Все слова в нем – мои, но все методики разработаны другим, причем вполне конкретным автором. Если он только пожелает, я с удовольствием размещу его фамилию на обложке этой книги как фамилию своего соавтора. Гонораром я уже предлагал делиться, но ответа не получил.

Руководство по руководству

Каждый подчиненный должен чувствовать, что он дерьмо.

Если он не чувствует этого, дайте ему это почувствовать всеми доступными вам средствами. Некоторые из них приводятся ниже.

1. Не давайте подчиненному получить удовлетворение процессом труда. Для этого проще всего дать ему невыполнимое задание или работу, рассчитанную на пять человек. Но это грубо и подрывает авторитет руководителя, поскольку демонстрирует его глупость или, в лучшем случае, неумение считать. Вот более приемлемые способы.

– Никогда не формулируйте четко, чем должен заниматься сотрудник. Если же он все-таки выдавил из вас перечень своих функций, немедленно поручите ему что-нибудь другое или, наоборот, поручите то же самое еще троим, причем желательно, чтобы сами исполнители об этом были не в курсе.

– Никогда не сообщайте сотруднику всю информацию, необходимую для выполнения задания.

Если же информацию ему все же удалось получить, немедленно дайте другому сотруднику, занимающемуся тем же самым, информацию, противоречащую первой.

– Весьма полезно объявить человека ответственным за что-либо на совещании, на которое вы его не пригласили. Самое главное после этого – дать ему понять, что он сам не пошел на совещание.

– Всегда старайтесь, чтобы никто не знал, кому он кроме вас подчиняется. Если вы не можете не подчинить одного сотрудника другому, обязательно подчините его сразу двоим, а самому подчиненному назовите в качестве руководителя кого-нибудь третьего. Интересны также схемы, в которых сотрудник оказывается подчиненным сам себе через одно или два звена.

– Очень полезно держать на работе хотя бы одного сотрудника, который в состоянии провалить любое порученное ему дело. Такой сотрудник должен выполнять обязанности, пересекающиеся или стыкующиеся с обязанностями наибольшего возможного количества других сотрудников. Очень вероятно, что в этом случае уже никто не сможет сделать что-нибудь по-человечески.

Приучайте подчиненных делать все через задницу. Когда вы заметите, что у них даже это начало получаться, поменяйте часть тела. Частей тела много, ко всем не приспособятся.

2. Не давайте подчиненному получить удовлетворение результатом труда.

– Никогда не принимайте окончательных решений и не назначайте окончательных сроков. Это можно делать только для того, чтобы отменить первое и передвинуть второе. Если сотрудник подготовил все заранее, передвиньте срок на более поздний или отмените задание, в противном случае передвиньте срок на более раннее время.

– Если успех какой-либо работы неминуем, можно поступить одним из двух способов. Или дайте сотруднику другое задание (еще лучше – отправьте в командировку) и торжественно завершите дело сами, или начинайте активно ему мешать: дайте его подчиненным другие задания (желательно требующие их присутствия в другом месте и так, чтобы руководитель об этом не знал), отберите обещанные ранее ресурсы (деньги, автомашину, связь и т. п.). Конечно же, все это

нужно делать в последний момент, когда изменить уже ничего нельзя.

3. Всегда показывайте, что вы умнее, сильнее, круче и лучше подчиненного.

– Если подчиненный принял какое-нибудь решение самостоятельно, обидьтесь, поругайтесь, повозмущайтесь и отмените это решение. Если он внес предложение, представьте его как невероятно бездарное, а если это не получается, скажите, что тут все надо обдумать. Обдумывание должно длиться не менее двух месяцев, после чего про предложение можно забыть или выдать его за свое. Если подчиненный не принимал решений и не вносил предложений, пожалуйтесь на его безынициативность.

– Никогда не извиняйтесь и не признавайте своих ошибок. Подчиненный виноват даже тогда, когда вы наступили ему на ногу.

– Никогда не пытайтесь выполнять свою работу: это может не получиться. Всегда делайте работу своих подчиненных, это беспроигрышный вариант: если получилось – «Мне приходится работать за всех», если нет – «Вам ничего нельзя поручить».

– Потопчите подчиненного. Лучше всего просто избить его или вымазать дерьмом, но это опасный и/или дорогостоящий путь. Еще лучше заставить его изнасиловать себя самого, но я не знаю, как этого добиться.

– Изумительно действует на подчиненных и такой маневр: дав сотруднику согласие переговорить с ним и предложив садиться, вы в ту же секунду начинаете набирать номер на своем телефонном аппарате и после соединения приступаете к получасовому общению с абонентом.

– В качестве очень эффективного средства рекомендуется организация переездов из одного помещения в другое. Самое главное при этом – проследить, чтобы условия работы ни у кого не улучшились.

– Никогда не показывайте, что вы довольны чьей-то работой. Лучше всего, чтобы сотрудник просто не мог понять, хорошо ли он работает. Для этого иногда даже стоит не обращать внимания на явные провалы в работе сотрудника (они и так видны). У вас всегда достаточно поводов поругать его за что-нибудь другое.

– Сотрудник никогда не должен знать, за что ему платят зарплату. Для этого иногда следует давать ему премию, чтобы снять ее в тот момент, когда ему все-таки удастся что-нибудь сделать.

– Выдавать зарплату лучше ближе к ночи, как минимум часа через два после окончания рабочего дня. При этом последние три часа нужно находиться на виду у всех и всем своим видом показывать, что у вас есть дела и поважнее, чем платить кому-то зарплату.

– Еще лучше иногда забывать отдать зарплату сотруднику. Идеальное время для этого – день зарплаты перед длительными праздниками или отпуском.

– После невыплаты двух получек подряд очень эффектно расплатиться за подчиненного за обед в столовой.

Если тем не менее ваш подчиненный так и не ощущает себя дерьмом, выгоните или, что еще лучше, выживите его с работы.

Теперь, когда все ваши подчиненные поняли, что они дерьмо, и сроднились с этой мыслью, вы можете начать ходить в белом фраке. Только сами позаботьтесь на всякий случай о следующем месте работы.

Кроме того, вот несколько правил, которыми я на самом деле пользуюсь.

1. Никогда не берите на работу родственников и друзей. У меня есть положительный опыт работы и с родственниками, и с друзьями, но проблемы, которые могут возникнуть после такого шага, заставляют меня продолжать придерживаться этого правила.

2. Сотрудник, который знает свое непосредственное дело хуже вас, – плохой сотрудник. Я исхожу из предположения, что вы собрались делать дело, а не тешить свое самолюбие.

3. Если качество или производительность труда программиста можно повысить с помощью материального стимулирования, то этого программиста следует уволить.

4. Качество и производительность труда программиста всегда можно понизить с помощью отрицательного материального стимулирования. Например, понижая ему зарплату после каждого внедрения, можно добиться полного отсутствия уже третьего внедрения.

5. Очень просто людей, преданных вам, превратить в людей, преданных вами. Чьи-либо успехи в обратных действиях мне неизвестны.

Еще одно правило, специфическое для разработчиков программного обеспечения. Я его изобрел во время внедрения первого проекта, но никогда им не пользовался. Хотя иногда очень хотелось. Называется оно «принцип модульности Орлова»: **если из распечатки одного модуля программы можно свить веревку, на которой можно повесить программиста, его написавшего, то все это следует сделать.**

Буратино в отделе

(Конспект лекции, прочитанный одному Буратино из отдела ИТ летом 2003 года)

Как ни печально, на работе люди используются утилитарно. Единственная задача, которую я выполняю в роли начальника, это техническое обслуживание, наладка и ремонт механизма под названием «отдел» для того, чтобы он мог выполнять свои функции. Люди при этом соответствуют деталькам, совместная работа которых как раз и приводит к тому, что механизм работает. Если с одной из деталек происходят какие-то изменения, я ее должен осмотреть, отремонтировать или изъять, после чего переналадить механизм так, чтобы он мог продолжить свою работу.

Поскольку я до некоторой степени человек и детальками в отделе являются тоже люди, то происходящее с детальками может затрагивать меня не только в роли начальника отдела, но и чисто по-человечески. Поэтому до, после и во время этих производственных операций я могу пить валокордин, не спать ночами и даже прекратить играть роль начальника вовсе, если необходимые ремонты и замены деталек слишком болезненны для меня как человека или противоречат моим моральным установкам. Но если я от роли еще не отказался, то я буду продолжать наладку механизма и пить валокордин.

В армии механизмы-подразделения строятся на принципе абсолютной взаимозаменяемости деталей, и это объективная необходимость, поскольку во время боевых действиях детали выходят из строя достаточно часто. Поэтому в процессе подготовки деталей их все сделают или одинаково круглыми в профиле, чтобы одинаково катились, или одинаково квадратными, чтобы можно было в любую часть стены вставить. Буратино в армии (подчеркиваю: в любой, а не только в нашей) лишится носа и других выступающих частей и нарастит шею, чтобы ничем не отличаться от других бревен. Иначе он будет сломан.

По счастью, в моем отделе это не так. Я могу несколько раз переставить деталь, пока она не встанет на место, где ее

взаимодействие с другими деталями будет наиболее эффективным для функционирования механизма в целом, могу даже подыскать нишу для длинного носа, прослежу, чтобы ручки и ножки не выломались из суставов при частых контактах с соседними арлекинами, но все это тоже имеет пределы. Если для носа такой длины ниши в отделе нет, то его придется укоротить. Туловище я обработаю шкуркой, чтобы окружающие мальвины не пострадали от заноз. В худших случаях придется заменить деталь на такую, которая не будет ломаться и приведет к повышению эффективности работы механизма. Я все сказал. Ты умный. Все остальное зависит от тебя.

Формулирование задач

В соответствии с российскими (и, кстати, американскими) стандартами составление технического задания (ТЗ) возлагается на разработчика. Это естественно, так как обычно клиент сам не знает, чего хочет, что можно реализовать и как.

Однако зачастую сотрудники разработчика если и умеют писать программы, то совершенно не в состоянии сформулировать свои мысли на русском языке. В качестве компенсации этого недостатка в договор на разработку вставляется фраза «Если какие-либо положения технического задания могут быть интерпретированы неоднозначно, правильной считается интерпретация, использованная разработчиком».

Посмотрим на примерах, к чему это приводит.

> Невинная фраза из ТЗ «Все количество распределяется по всем покупателям поровну с округлением до 1 шт. Погрешность округления относится на последнего покупателя» приводит к очень интересным результатам.
>
> Итак, на 100 покупателей поступило 40 штук. Поровну означает 40/100 = 0,4 штуки. С округлением до одной штуки получается 0, то есть все 40 штук получит последний покупатель.
>
> Еще веселее, если на 100 покупателей поступило 60 штук. Тогда все покупатели, кроме последнего, получат по одной штуке, а последний… – 39 (минус тридцать девять) штук.
>
> Программа соответствует ТЗ.

Вообще ошибки округления – это бич современных систем учета, особенно в России, где правила бухгалтерского учета про них просто не упоминают, якобы у нас все всегда точно. Как следствие, в России их никаким способом нельзя обрабатывать правильно. У меня сформировалось стойкая убежденность, что наши налоговые органы вполне умышленно никогда не отвечают на такие вопросы и не дают необходимых разъяснений: так удобнее сделать нарушителем любого, кто отчитывается перед налоговиками. Следующий пример из приказа МНС я могу воспринимать только как издевательство:

> «По коду строки 100 указано 5 месяцев, в течение которых транспортное средство было зарегистрировано

на организацию в 2003 году, следовательно, коэффициент, который следует указать по коду строки 110, составит 5/12 = (5: 12)».

Приказ МНС № БГ-3-21/724 от 29 декабря 2003 г.

Ясно? 5/12 равно 5:12, так МНС приказал. А уж сколько знаков после запятой оставить – это ваша проблема: сколько ни укажите, будете не правы.

Значит, тем более все вопросы, связанные с округлениями, надо подробно описать в техническом задании и по возможности разъяснить руководству (с примерами на конкретных цифрах).

Про программиста, написавшего программу для банка, добавлявшую все ошибки округления на его банковский счет и ставшего миллионером (на время, пока его не посадили), я читал еще в начале семидесятых (книжка называлась «Кибернетическая смесь»). Теперь обязательно пытаюсь рассказать эту историю боссам, а потом уже показываю, как после получения груза на сто тысяч долларов тысяча долларов растворяется в системе.

Мнимые очевидности

Одним из самых крупных подводных камней разработки ТЗ является недоформализация тех задач, которые кажутся совершенно очевидными бизнес-заказчику. По всей видимости, нужно иметь специальную голову, чтобы всегда понимать, что записанные требования нельзя запрограммировать однозначно.

Про полтинники. Я уже писал, как в компании, продававшей периодическую печать, киоскеры, работавшие в метро, взмолились, чтобы мы округляли розничные цены на газеты до полтинника. Руководство нас на это благословило, и я программисту ровно так задание и сформулировал: «В поле PriceMetro поместить значение из поля PriceRetail, округленное до 0,5».

Звонит программист:

– А каким способом округлять?

– Обычным.

– Андрей, я не знаю *обычного* способа округлять до 0,5. Пожалуйста, опиши.

– Ты издеваешься?

– Нет, я серьезно.

– Хорошо.

Я в раздражении швыряю трубку, открываю текст задания и… задумываюсь. На всякий случай лезу в Интернет, потом благодарю Бога, что у меня есть умный и спокойный программист, и записываю в задании:

```
ДРОБН:= А – ЦЕЛОЕ (А)
ЕСЛИ ДРОБН < 0,25 ТО ДРОБН:= 0
ИНАЧЕ ЕСЛИ ДРОБН < 0,75 ТО ДРОБН:= 0,5
ИНАЧЕ ДРОБН:= 1
ОКРУГЛ05 (А) = ДРОБН + ЦЕЛОЕ (А)
```

Дело в том, что не существует никакого «общего» способа округления до полтинников. Его нужно было записать в задании явно. Что я и сделал самым понятным для программиста способом.

Про дополнительное оборудование (история, рассказанная Александром Ройзнером за 15 лет до предыдущей). Мы пытались формализовать действия, которые во время поездки должен совершать машинист локомотива, чтобы в дальнейшем правильность этих действий оценивала автоматизированная система. По этому поводу с начальником одного из локомотивных депо произошел следующий диалог:

– По правилам при проверке тормозов нужно после появления тормозного эффекта и снижения скорости на 10 км/ч в грузовом груженом, грузопассажирском, пассажирском поезде и на 4 – 6 км/ч в грузовом порожнем поезде произвести отпуск тормозов. Снижение скорости мы определить можем. А как определить «эффект торможения»?

– А эффект торможения каждый машинист задницей чувствует.

– Значит, нам надо включить в измерительную систему задницу?

Кстати, прекрасные образцы для подражания в постановке задач имеются в Книге Левит Ветхого Завета:

«скажите сынам Израилевым: вот животные, которые можно вам есть из всего скота на земле:

всякий скот, у которого раздвоены копыта и на копытах глубокий разрез, и который жует жвачку, ешьте;

только сих не ешьте из жующих жвачку и имеющих раздвоенные копыта: верблюда, потому что он жует жвачку, но копыта у него не раздвоены, нечист он для вас;

и тушканчика, потому что он жует жвачку, но копыта у него не раздвоены, нечист он для вас,

и зайца, потому что он жует жвачку, но копыта у него не раздвоены, нечист он для вас;

и свиньи, потому что копыта у нее раздвоены и на копытах разрез глубокий, но она не жует жвачки, нечиста она для вас» (Левит 11: 2–7).

Вот настоящий пример Божественной формализации: сначала описано общее правило, а потом еще дано несколько примеров, подтверждающих, что в общем правиле условия, подлежащие проверке, соединены логическим «и», а не «или».

Оставьте здравый смысл дома

Обычно постановщики-фантазеры гнездятся в компаниях-разработчиках, но иногда встречаются и внутри производственных и торговых предприятий. Бойтесь их.

При описании бизнес-процессов фантазия постановщика становится очень большим недостатком. Правило здесь есть только одно, но оно не допускает отклонений.

Если какие-то части бизнес-процесса вам не известны точно, то их нужно узнать, а не выдумать.

Здравый смысл и логику в такие моменты лучше оставлять дома. Иначе в системе будут реализованы решения красивые, логичные, но абсолютно не соответствующие реальности.

«По логике вещей это должно быть так». По логике, может, и так, но кто вам сказал, что бизнес-процессы подчиняются хоть какой-то логике? А если логика и есть, то она совершенно не обязана быть айтишной, аристотелевой или даже женской. Самое ужасное, что бизнес-процесс создается совместными стараниями людей, может быть, и разумных, но преследующих самые разные цели. Есть закон, основанный на думной логике, подзаконные акты, основанные на смеси нескольких ведомственных логик (к примеру, налоговой, милицейской и железнодорожной), бизнес-правила ваших крупных контрагентов, которые вы не в состоянии поменять или не применять, жгучее желание вашего хозяина, противоречащее не менее жгучему желанию его компаньона, а напоследок ограничение по количеству кнопок, на которые младший бухгалтер может нажать, не ошибаясь слишком часто.

Вот может измениться сумма денег в зависимости от количества накладных, по которым продан товар? «По логике вещей» нет.

А теперь возьмите карандаш и запишите варианты, в которых это не так, исходя из вашего опыта, не поглядывая в то, что я написал ниже.

Теперь давайте сравним.

Округлениестоимости по каждой из двух накладных даст не тот же результат, что округление по одной накладной.

Фиксированная сумма на накладную: за оформление и/или доставку одного заказа берется фиксированная плата. Если накладных две, то эту сумму возьмут дважды.

Скидки, зависящие от позиции товара в накладной. Правила розничной продажи могут включать скидки, зависящие и от общей стоимости покупки, и от количества предметов в покупке, и даже от порядка следования строк в накладной («при покупке на сумму свыше 1000 рублей скидка 3 %», «купите два товара и получите третий бесплатно», «на второй товар скидка 5 %, на третий – 10 %, на четвертый и последующие – 15 %» и т. д.).

Рассчитываемая стоимость доставки: в сумму накладной включена стоимость перевозки товара. Тут Кафка отдыхает.

Вот выдержка из правил расчета стоимости перевозки мелких отправок железнодорожным транспортом:

> *«Минимальный расчетный вес отправки 10 кг. При исчислении провозной платы масса груза до 1000 кг округляется до полных 10 кг, а масса груза свыше 1000 кг округляется до полных 100 кг.*
>
> *Количество груза определяется в м³*, если масса 1-го м³ груза менее 200 кг.
>
> *Объем груза округляется до 0,05 м³».*

Теперь вам понятно, что стоимость доставки 6 кг груза по двум накладным будет ровно в два раза больше, чем по одной, а вот стоимость доставки 1010 кг по одной накладной будет больше, чем по двум накладным на 500 и 510 кг.

Еще интереснее случай перевозки пенопластовых блоков и чугунных гирь. Включив и то и другое в одну накладную, вы заплатите за вес перевозимого, а оформив две накладные, за чугун будете платить по весу, а за пенопласт – по объему.

А теперь рассмотрим еще один вопрос.

Может ли быть так, чтобы за два одинаковых автомобиля с одинаковыми двигателями, отличающихся только выбитыми на их отдельных частях номерами и находящихся в собственности и распоряжении у одного лица, нужно было платить разный транспортный налог?

Вот реальный случай. В одной партии грузовиков с завода поступили два одинаковых «КамАЗа». В их паспортах была указана мощность двигателей 184 киловатта (а в лошадиных силах мощность не указана). В паспорте одного из грузовиков

оказалась ошибка в номере шасси. Грузовик остался на месте, а паспорт отправили на завод переписывать. Новый паспорт выписывали более тщательно и мощность двигателя указали не только в киловаттах, но и в лошадиных силах: 250 л с. (184 кВт).

Все дальнейшие фокусы связаны с тем, что Инструкцию по заполнению паспортов транспортных средств создало и утвердило Министерство внутренних дел и в ней нет правил округления. Поэтому завод вправе написать в паспорте то, что написал. А вот Методические рекомендации по применению главы 28 «Транспортный налог» части второй Налогового кодекса Российской Федерации разработало и утвердило Министерство по налогам и сборам. И там в разделе V, п. 19 читаем:

> *«В случае если в технической документации на транспортное средство мощность двигателя указана в метрических единицах мощности (кВт), то соответствующий пересчет во внесистемные единицы мощности (лошадиные силы) осуществляется путем умножения мощности двигателя, выраженной в кВт, на множитель, равный 1,35962 (переводной коэффициент – 1 кВт = 1,35962 л с.) (Физические величины: Справочник / А. П. Бабичев, Н. А. Бабушкина, А. М. Братковский и др.; под ред. И. С. Григорьева, Е. З. Мейлихова. – М.: Энергоатомиздат, 1991.)*
>
> *При этом при пересчете во внесистемные единицы мощности (лошадиные силы) округление производится с точностью до второго знака после запятой».*

Аминь. Вот фрагмент таблицы налоговых ставок.

Наименование объекта налогообложения	Налоговая ставка (в рублях)
Свыше 200 л.с. до 250 л.с. (свыше 147,1 кВт до 183,9 кВт) включительно	35
Свыше 250 л.с. (свыше 183,9 кВт)	45

Смотрим таблицу и обнаруживаем, что грузовик, у которого мощность двигателя указана только в киловаттах, попадает в строку «Свыше 250 л с.» со ставкой 45 рублей за лошадь. Переводим киловатты в лошадиные силы: 184 х 1,35962 = 250,17008, округляем до второго знака и умножаем на ставку.

Налог за 12 месяцев получится равным 250,17 х 45 = 11257,65 рубля.

Для грузовика, у которого мощность указана в лошадиных

силах, ставка берется из строки «до 250 л с. включительно». Налог за 12 месяцев оказывается 250 x 35 = 8750 рублей.

Разница в налоге для одинаковых грузовиков составила 2507,65 рубля. Неплохо, а?

Практическим следствием этого замечательного факта становится то, что информацию о транспортном налоге в номенклатурном справочнике можно хранить только для приблизительных оценок и прогнозов. А вот информацию для точного расчета налога нужно хранить для каждого автомобиля отдельно.

Предупреждение. Пожалуйста, не забудьте, что я написал этот текст не в тот момент, когда вы его читаете. За время, прошедшее между этими событиями (хотя Событие, конечно, только то, что вы эту книгу читаете, а не что, я ее написал) многие нормативные акты по налогообложению могли неоднократно поменяться, а во многих программах одни ошибки исправили, а другие добавили. Не используйте эту книгу ни в качестве пособия по расчету налогов, ни в качестве руководства по каким-либо программным средствам.

«Мне так нужна распределенность!»

Хочется обратить внимание читателя на один вопрос, который должен быть решен в самом начале проекта. Это вопрос о том, где хранятся, где обрабатываются и где используются данные. В зависимости от ответа результатом проектирования станут решения, принципиально отличающиеся друг от друга на уровне архитектуры.

Традиционная ошибка, которую разработчики совершают довольно часто, состоит в том, что решение этого вопроса откладывается «на потом» в надежде, что превратить локальную систему в распределенную удастся с помощью небольших модификаций. Я сталкивался с системами, реализованными таким способом, на этапе их сопровождения. Сталкивался очень больно. В системах постоянно появлялась информация, перекореженная в результате активной работы пользователей, разделенных тысячами километров и несколькими часовыми поясами, а я каждый день должен был собирать информационные пазлы, не только рассыпавшиеся, но еще и сломанные в процессе транспортировки, в каждом случае пытаясь понять, в каких местах нашей необъятной Родины и в какой последовательности были нажаты кнопочки и какие скрипты теперь нужно написать, чтобы восстановить сколько-нибудь непротиворечивое состояние

системы.

Ситуация с проектированием распределенных систем вдобавок сильно затуманена большим количеством специальных терминов, сильно пугающих неспециалистов, а специалистами трактуемых совершенно по-разному. Поэтому я сначала попытаюсь сформулировать на относительно простом русском языке основные выводы для читателей, для которых специальные слова типа «коллизия», «клинч», «асинхронная репликация» мало что значат, а потом попытаюсь объясниться на специальном жаргоне с более смелыми.

Проще и удобнее всего хранить и обрабатывать информацию в одном месте, а потом передавать ее пользователям (как людям, так и другим информационным системам), расположенным во всех концах земного шара, по каналам связи. В этом случае минимизируется количество ошибок, связанных с одновременными попытками пользователей изменять информацию в системе. На компьютерах пользователей, где бы они ни находились, должно быть установлено только программное обеспечение, позволяющее принимать, отправлять и отображать информацию (программа-клиент). Сейчас появилось много систем, в которых пользователю для начала работы необходимо иметь только обычный интернет-браузер, а необходимые для работы скрипты загружаются по необходимости с сервера (тонкий клиент). Современные каналы связи в большинстве случаев позволяют работать с информационными системами таким способом.

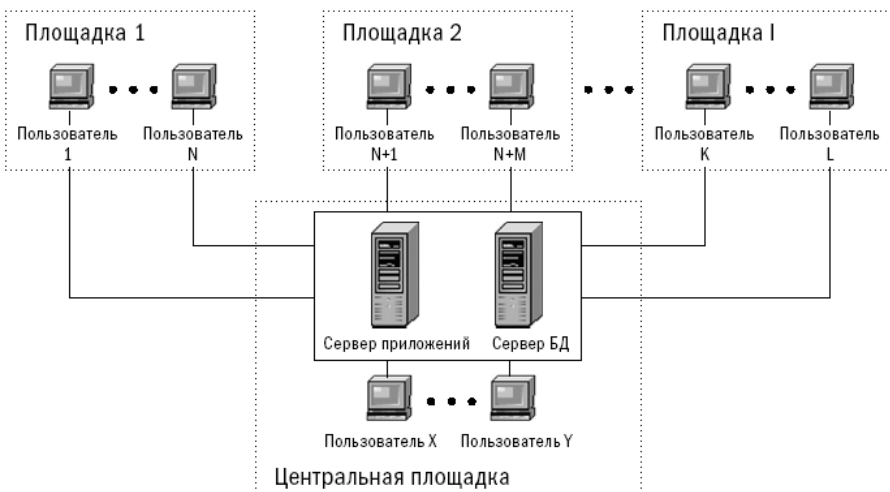


Рис. 7. Пример системы, работающей с удаленными пользователями

Все остальные способы работы следует использовать только в

случае невозможности с нужной скоростью обрабатывать и передавать по каналам связи необходимые объемы информации в течение всего времени функционирования системы.

В гипер-суперсистемах возможны ситуации, когда для хранения и обработки информации не хватает мощностей серверного оборудования, расположенного на одной площадке, но каналы связи обладают достаточной мощностью и надежностью. В таких случаях используются синхронные распределенные системы (иначе – системы с синхронной репликацией). На уровне бизнес-логики распределенность таких систем не видна: как вас не слишком волнует, расположена база на одном диске или на многих, на одном сервере или на многих, так вам не очень важно, расположена база в одном Урюпинске или еще и в Арзамасе. И если пользователь в Арзамасе начал редактировать запись о контрагенте, то другому пользователю из Урюпинска просто придется подождать, поскольку эта запись будет заблокирована.

> Собственно, блокировки будут и при использовании централизованной базы данных. Пресловутый онлайн – мечта всех управленцев – означает всего лишь уменьшение задержки при передаче информации. Кстати, если определить, при передаче какой информации нужно минимизировать эти задержки, может выясниться, что это далеко не весь массив данных. Если еще начать выяснять, какова же все-таки максимально допустимая задержка, то может оказаться, что необходимо, например, иметь актуальную информацию об остатках товара один раз в день, а все остальное достаточно обновлять раз в неделю. Так мы приходим уже к периодической репликации.
>
> Конечно же, бывают случаи, когда задержка даже в секунды может дорого стоить (классический пример – биржевая торговля), но чаще всего управленец, требующий непременно данных в режиме реального времени, не будет ими пользоваться чаще одного раза в сутки. И возможно, эти данные ему можно предоставить и более дешевыми средствами, нежели прокладкой каналов до каждого ларька и установкой многомиллионной системы на рабочем месте каждой уборщицы. Понимание этого позволяет значительно сэкономить средства – именно так получаются вполне успешные внедрения маломощных систем в огромных компаниях. – *Д. К.*

Именно системам с синхронной репликацией посвящено наибольшее количество теоретических работ по распределенным системам. Но такие системы страшно дороги и относительно редки, и, если вы работаете не в корпорации уровня Газпрома, а всего лишь в небольшом холдинге, включающем пару-тройку заводов и сотню магазинов, то вряд ли вам это нужно.

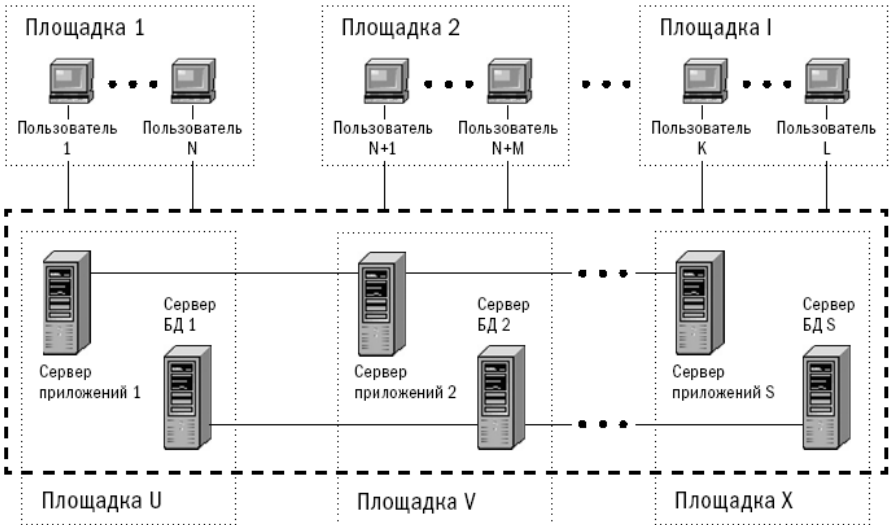


Рис. 8. Пример синхронной распределенной системы

Основные проблемы с распределенной системой начинаются, когда производительности каналов не хватает, чтобы данные обрабатывались удаленно, а результаты посылались клиентам, или каналы связи не слишком надежны, а с системой нужно продолжать работать на всех площадках, даже когда каналы не работают из-за повышенной солнечной активности, пьяного экскаваторщика, перерубившего кабель, или урагана.

Собственно, не так давно «каналом связи» между двумя площадками в Москве в компании, где я работал, служил курьер, который перевозил винчестер с изменениями базы данных с площадки на площадку два раза в день.

В таких случаях пользователи на каждой площадке работают со своей местной (локальной) базой данных, а сами базы выравниваются или периодически, или когда возникает такая возможность.

Урюпинский пользователь в такой системе может хладнокровно править наименование организации в соответствии с ИНН в тот самый момент, когда арзамасец начнет изменять в своей копии той же записи ИНН организации в соответствии с названием. Эта шутка носит высоконаучное название коллизии.

Понятно, что попытка свести воедино арзамасско-урюпинские результаты (то есть разрешить коллизию) вызовет определенные проблемы.

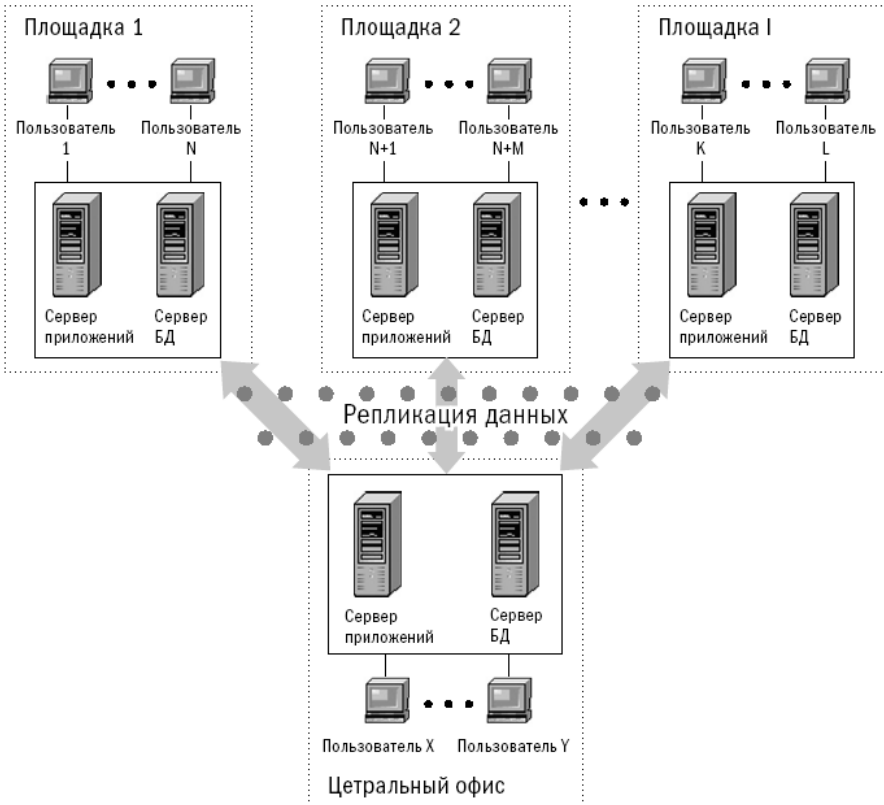


Рис. 9. Пример распределенной системы с асинхронными репликациями

Такие системы называются системами с асинхронными репликациями, а сам комплект изменений, за один раз передаваемый от одной локальной базы к другой, – репликой. Те, кому не нравится импортное слово «репликация», используют слово «тиражирование», которое почему-то считают более русским. На мой взгляд, больше подходит «выравнивание», но не хочется путать читателя еще больше.

Механизмы корректной работы систем с асинхронными репликациями наиболее сложные, и, как я уже писал, если они не заложены в проект и не продуманы на начальной его стадии, все заканчивается достаточно плохо.

Сейчас любая уважающая себя СУБД имеет встроенные механизмы, которые позволяют организовывать асинхронные репликации и настраивать способы разрешения коллизий. Многие на это ловятся, считая, что «в случае чего» это дает возможность

быстро превратить нераспределенную информационную систему в распределенную.

Не тут-то было. Выясняется, что необходимость обмениваться данными нужно было в процессе проектирования держать в голове постоянно, разрабатывая как схему базы, так и способы обработки данных.

Например, из реплицируемой базы данных ничего нельзя удалять физически, поскольку после этого вы потеряете и информацию о том, что нужно удалить из других локальных баз. Если вы так поступали, то это придется поменять.

Очень интересно получится, когда сервера системы начнут функционировать в разных часовых поясах. Конечно, СУБД при репликации позаботится о модификации временных меток, которые она сама установила, но как работать с другими датами и временами, записанными в базе, вам было бы неплохо заранее разобраться самому, чтобы потом не гадать, глядя на 0:00, была в этот момент полночь в Москве или в Петропавловске-Камчатском.

Самые большие проблемы могут возникнуть при передаче взаимосвязанной информации, поскольку в принимающей базе в общем случае этой информацией можно пользоваться только после того, как в нее попадут все взаимозависимые записи.

> **Пример.** Пусть по каким-то причинам вы храните фамилию в одной таблице базы, а имя и отчество – в другой, ссылающейся на первую. В Урюпинске оператор заполнил форму, в которой указал Орлв Андрей Георгиевич. С помощью репликаций эта информация должна оказаться в Арзамасе. Реплики отправляются по расписанию, объем одной реплики ограничен. Поэтому в Арзамас в первой реплике пришла только фамилия из главной таблицы. Оператор в Арзамасе увидел ошибку в фамилии и исправил ее. Теперь в Арзамасе в главной таблице «Орлов», а в зависимой еще пусто. Наконец со следующей репликой в Арзамас приходит запись, в которой указано «Андрей Георгиевич». Но в базу она не попадет: главная запись уже была изменена, и процедура автоматического разрешения коллизий «Андрея Георгиевича» отвергнет.

Если же вы еще и не все зависимости описали на уровне СУБД, а поддерживаете какие-то из них с помощью алгоритмов обработки, то становится совсем загадочным, как это все собрать на принимающем конце.

А поскольку многие системы стараются

поддерживать несколько СУБД, то вероятность использования механизмов приложения, а не СУБД в них достаточно велика, что делает репликацию средствами СУБД достаточно сложной. – *Д. К.*

Арсенал средств обработки в современных СУБД весьма внушителен, все они полезны в каких-то случаях, но желание разработчика использовать все эти средства сразу может приводить к результатам слабо предсказуемым. А ведь часто данные в нескольких зависимых записях сначала изменяются хранимой процедурой, в работу которой вклиниваются несколько триггеров, после чего результат полируется заданием (джобом), который запускается каждые пять минут, а посреди всего этого отрабатывает репликация, а потом нечто похожее происходит и в базе-приемнике…

Продолжать скорбный список можно достаточно долго, но я надеюсь, что основную мысль я все-таки донес: попытка превратить неплохо работающую систему с нераспределенной базой в распределенную систему может привести к необходимости делать проект сначала.

Еще немного про коллизии, то есть про независимое изменение одних и тех же данных на разных площадках. При репликации они должны быть разрешены, а значит, из двух вариантов изменения должен быть выбран только один или отброшены оба изменения.

Коллизия – это в любом случае плохо. Разрешение коллизии, каким бы способом оно ни производилось (по времени изменения записи, по приоритету площадки или пользователя, администратором системы вручную после необходимых телефонных звонков), означает, что вы похоронили чью-то работу.

Поэтому основные усилия на начальной стадии проекта, безусловно, надо направить на попытки устранения коллизий регламентным путем, разделив места создания и изменения информации или хотя бы время таких действий.

Коллизий не будет, если вам удастся организовать работу так, чтобы на разных площадках изменялись разные типы объектов. Проблем не будет, если номенклатуру и контрагентов добавлять только в центральном офисе, а использовать на всех площадках.

Несколько хуже, когда на разных площадках изменяются разные объекты одного типа, различаемые по какому-то признаку. Хуже потому, что этот признак тоже можно на какой-то площадке изменять. То есть формирование приходных и расходных накладных на один и тот же товар не доставит вам проблем,

поскольку пара «Получатель» и «Отправитель» в таких накладных будет для каждой площадки разной, а вот с информацией о клиентах, приписанных к точкам обслуживания, работать будет хуже, поскольку клиенты любят менять точки обслуживания, и если вы отбирали клиентов для репликации по этому признаку, то неплохо понять заранее, где именно разрешить изменять этот признак и как будет происходить репликация измененной записи.

Для банковских и кассовых документов больше подходит регламентное разделение времени изменения документов. Например, три дня их можно изменять только на площадке, где они введены, а начиная с четвертого дня – только в центральном офисе.

Только вот отнюдь не всегда удается организовать работу так, чтобы это было удобно проектировщику информационной системы. Если руководство компании решит, что клиента можно обслуживать любым способом на любой площадке, вам придется поломать голову, как с наименьшим ущербом именно так и сделать.

Очень жаль, что в техническом задании не принято писать «Программа должна работать»…

Единственный общий совет этого раздела: старайтесь не экономить время на этапе постановки (написания технического задания). Реально это единственный этап, на котором время вам выделено именно для того, чтобы вы думали – дальше нужно будет трясти. Правда, руководство и на этом этапе требует сверхскоростей. Ведь для него хуже нет, чем вот уже два месяца платить зарплату и не иметь с этого не только никакого барыша, но даже хотя бы какого-то видимого результата. Однажды на очередной покрик «Давай-давай!!!» я отыскал в Интернете, распечатал и раздал всем генерально-исполнительным начало первой главы «Винни-Пуха»:

> *«Ну вот, перед вами Винни-Пух.*
>
> *Как видите, он спускается по лестнице вслед за своим другом Кристофером Робином, головой вниз, пересчитывая ступеньки собственным затылком: бум-бум-бум. Другого способа сходить с лестницы он пока не знает. Иногда ему, правда, кажется, что можно бы найти какой-то другой способ, если бы он только мог на минутку перестать бумкать и как следует сосредоточиться. Но увы – сосредоточиться-то ему и некогда».*

Алан Александр Милн «Винни-Пух и все-все-все» (пересказ Б. Заходера)

Авторитет Милна на время подействовал.

Траблы-грабли-бумс!

Личный опыт не всегда может использоваться в качестве критерия истинности.

Среди занимающихся альпинизмом многие могут вам рассказать, как они сами нарушали правила безопасности во время восхождений и с ними ничего не произошло. Обратным опытом никто не делится. Покойники вообще не склонны делиться опытом.

Информационные системы разрабатывают уже более пятидесяти лет, и определенный опыт в этом деле накоплен. Не следует в его анализе исходить из предположения, что до вас, такого крутого и гениального, информационными технологиями занимались только идиоты. Второе предположение, что все правила придумали из-за недостатка быстродействия и памяти, а сейчас все по-другому, тоже ложно.

Казалось бы, перечисленные ниже утверждения давно стали аксиомами, так ведь все время находятся лобачевские, которые полагают, что они без этих аксиом обойдутся. И каждый раз получают по лбу. Поэтому все-таки не пренебрегайте очевидным нижеизложенным.

– Любую систему придется изменять на всех этапах ее жизненного цикла.

– Любые обещания, что что-то не будет меняться, а уж тем более расти в объеме, никогда не выполняются.

– В больших системах не бывает маловероятных событий. Все, что может случиться, обязательно случится. Пользователи в России и Австралии обязательно нажмут на нужные кнопки одновременно, если вы этого не предусмотрели.

– Если одна и та же информация хранится в двух местах системы, то через неделю эксплуатации в этих местах будет разная информация.

– Нарушение известных правил проектирования с целью экономии времени разработки всегда приводит к увеличению времени разработки.

Я совсем не считаю, что свод правил проектирования вечен и неизменен, но если вы хотите какое-то из этих правил нарушить, то сначала поймите, почему оно появилось и какие суперпреимущества вам дает его отмена.

Еще одна очевидность, про которую забывают разработчики: удобство и скорость разработки, безусловно, важны, но все-таки система в итоге оценивается по ее потребительским свойствам.

Если есть выбор, написать семь процедур или одну, но в последнем случае скорость отклика системы изменится с 0,5 с на 5 с, то писать надо семь процедур.

Впрочем, уверен, что этот раздел написал без толку. Похоже, это тоже одно из вечных правил.

Ниже приводится перечень грабель, на которые наступали и продолжают наступать настолько часто, что их рукоятки уже отполированы лбами проектировщиков.

Грабли-рекордсмены

Наверное, рукоятка таких грабель изготовлена из особо ценных и прочных пород дерева. Иначе непонятно, как они столько лет бьют по лбам горе-проектировщиков и до сих пор не сломались.

Сейчас мне кажется, что абсолютный запрет на любую семантическую интерпретацию идентификационных кодов мне внушили еще в детском саду. Вряд ли это так. Но по окончании института (хотя это произошло через пятнадцать лет после окончания детского сада, но уже тридцать лет назад) я точно знал, что в идентификационном коде не должно быть никакой семантики.

Никакой. НИКАКОЙ! **НИКАКОЙ!!!** Только циферки, которые однозначно определяют объект, и никакого другого смысла. Этому учили еще до появления классических работ Дейта и Кодда.

Но все-таки в каждом проекте у кого-нибудь начинает зудеть и чесаться.

«А давайте первая цифра кода у нас будет номером цеха, который выпускает изделие, вторая – номером отдела, который занимается продажей…» И что произойдет, когда у вас появится десятый цех или отдел? А когда изделие начнут в двух цехах собирать, какую цифру ставить? И как перевести производство из одного цеха в другой? Менять код изделия или запрещать перевод производства?

> Да, а еще в код иногда засовывают буквы. А буква «А» есть как в русском алфавите, так и в латинском. И – увы! – для системы это РАЗНЫЕ буквы (хотя и не всегда можно это объяснить пользователю). Это создает много радости при ручном вводе новых идентификаторов и еще больше радости, если справочники и транзакции приходится загружать из другой системы. – *Д. К.*

«А давайте включим в id товара код группы и подгруппы…» А что вы будете делать, когда потребуется поменять код группы или подгруппы? Лопатить всю базу, разыскивая нужный id, и заменять его на другой? «А мы ничего не будем делать. Все равно в 90 % случаях группа и подгруппа будет верной». Вы этой мыслью с главбухом поделитесь. Ему очень понравится правильный расчет НДС с вероятностью 0,9. И не исключено, что на одного плохого проектировщика станет меньше.

«А давайте у нас id накладной и будет ее номером». Не давайте. Когда система встанет, накладные будут выписывать вручную, а потом в систему нужно будет ввести именно те номера, которые оказались на бумажных накладных, даже если это одинаковые номера. И накладные поставщиков неплохо завести в систему под теми номерами, под которыми они нам выданы.

И выучите, наконец, зазубрите, напишите на мониторе, сделайте татуировку на лбу: **вводя любое информационное поле для любого объекта, можно ошибиться или передумать. Поэтому идентификационный код всегда—всегда!!! – должен генерироваться автоматически и не иметь никакой информационной нагрузки.**

Грабли классические

Суббота, десять вечера. Из продовольственного магазина, в котором внедряется пилотный проект складской системы, звонит кладовщик. Едва не плача, он сообщает, что принял товар на склад, а секции его выдать не может, поскольку товар пропал. А если его срочно не продать, то он протухнет.

Приезжаю. Пытаюсь найти товар на складских карточках. Действительно, найти не получается. А приходная накладная есть. В ней все правильно. Смотрю отчет по товародвижению – в нем все на месте. Проверяю суммарные остатки по складу – накладная учтена. То есть товар на складе есть, а вот найти его нельзя. Правда, на этом складе уже восемь тысяч карточек, так что визуально товар не обнаружить, если даже он есть. Уточняю, что делал кладовщик.

– Это новый товар. Я его сначала завел по бумажной накладной в справочник номенклатуры, потом ввел саму накладную. А потом посмотрел на сам товар и понял, что это тефтели, а не котлеты, как написано в накладной.

– И как ты поступил?

– Исправил запись в справочнике номенклатуры. А в накладной все изменилось само.

– Кажется, я понял, – отвечаю я, подумав. – Смотри, – и набираю в строке поиска «котле». Экран прокручивается, и на нем подсвечивается строчка «Тефтели». За спиной раздается восхищенное «Ой», но я в это время уже представляю, что я сделаю с разработчиками в понедельник.

Но воскресенье, предшествующее встрече, их спасает. Я не только никому ничего не пытаюсь оторвать, я даже могу разговаривать, используя только слова литературного русского языка:

– Вы что, название товара записываете на складской карточке?

– Мы только первые пять символов записываем, чтобы поиск шел быстрее.

Тут я становлюсь даже ласковым:

– А что вы делаете, если складские карточки уже созданы, а я наименование в номенклатурном справочнике поменяю?

– Кажется, ничего не делаем.

– То есть если я завел на склад «котлеты», а потом исправил в справочнике номенклатуры название на «тефтели», то что я увижу на карточке?

– «Тефтели».

– А искать мне что нужно?

– «Котлеты»… Наверное, это не совсем правильно…

– *Не совсем правильно?*

– Согласен, это совсем неправильно. Мы к следующему обновлению переделаем…

Я начал именно с примера, чтобы не пугать читателя высоконаучными словами. Но в приведенном случае были нарушены принципы проектирования баз данных, описанные в классических работах 1970-х годов. Таблица базы «Складские карточки» не находится в третьей нормальной форме. Про это уже столько понаписано, что мне и добавить нечего. Появление новых СУБД и новых способов поддержания зависимостей в базе данных совершенно ничего не меняет: грабли продолжают работать даже при использовании триггеров и джобов (заданий, запускаемых автоматически в определенные моменты суток или через определенные временные интервалы).

Все попытки поддерживать целостность и

непротиворечивость данных не на уровне схемы базы, а с помощью программных примочек натыкаются на одно практически непреодолимое препятствие: в достаточно сложных системах вы просто забываете это сделать для некоторых вариантов работы.

Кстати, неумелое использование перечисленных инструментов, на мой взгляд, приводит к последствиям более страшным, чем их полное неиспользование.

Грабли ленивые

Вы задумывались, что произойдет, если джоб, который стартует каждые 15 минут, в среднем работает два часа? А что будет, если выполнение триггера займет две минуты? В обоих случаях последствия могут быть разнообразными, но всегда неприятными и, что еще хуже, непредсказуемыми.

Не знаю, влияет ли на это постоянное увеличение быстродействия компьютеров, но почему-то очень-очень многие программисты предполагают, что код, который они написали, будет выполнен мгновенно. Последовательность этих «мгновений», воплощенная в информационной системе, заставляет пользователей минутами дожидаться хоть какой-нибудь реакции на нажатие кнопок. Но хуже другое: в конфликт начинают вступать процессы, о взаимодействии которых никто не подумал, потому что все мыслили эти процессы мгновенными.

Грабли феодальные

Если в системе появляются элементы документооборота, то документы (проекты договоров, заявки на материальное снабжение, предложения по изменению справочников и т. п.), созданные или исправленные одним пользователем, необходимо отправлять на согласование другим пользователям. Стадию прохождения документа по цепочке обычно называют статусом документа. Цепочки таких согласований бывают достаточно длинными. Естественно, встает вопрос, кто должен увидеть и кто обработать документ, находящийся в соответствующем статусе.

Поскольку в учетной системе этот вопрос не самый главный, разработчик иногда решает его с наскока и наиболее простым способом: адресация документов и права доступа к ним настраиваются по логинам пользователей, то есть в цепочку согласований включаются физические лица.

В результате все проблемы заменяемости сотрудников и изменения штатного расписания оказываются практически неразрешимыми: все перенастройки приходится выполнять вручную. Если этого не делать, то документ, отправленный на согласование по подчиненности, окажется у начальника, уволенного месяц назад, а не у того, кто стал исполнять его обязанности.

Феодализм все-таки давно пройденная формация. И если начальник управления уволился, то все отделы, входящие в управление, станут подчиняться новому начальнику, которого назначат, а не ходить в гости к старому, чтобы решать производственные вопросы. Поэтому при настройке цепочек согласования и утверждения документов привязываться нужно к функциональным ролям сотрудников или к должностям в штатном расписании, а не к физическим лицам.

Грабли демократические

Традиционный ляп разработки связан со способом назначения прав доступа на просмотр, создание и изменение объектов информационной системы. Обычно этот вопрос остается на периферии круга решаемых задач и, как следствие, решается уже после создания основного функционала системы, то есть ровно тогда, когда решить его практически невозможно. В довершение разработчик часто вообще не понимает, зачем все это нужно, и вводит разграничение доступа только под давлением заказчиков системы, да и то как бог на душу положит.

В результате в системе доступа образуются зияющие дыры. Наиболее традиционная дыра состоит в предположении, что вновь заводимому в систему пользователю позволено все, пока ограничения доступа не будут прописаны в явном виде. То есть в системе «разрешено все, что не запрещено». Но использование этого демократического принципа по отношению к информационным системам категорически противопоказано.

При заведении нового пользователя в систему его права должны быть пусты. Только после задания роли у пользователя появляются права в соответствии с этой ролью.

> Права, контрольные механизмы и прочие средства информационной безопасности (ИБ) – отдельная история при разработке и внедрении всех систем. Мало того что требования ИБ не принято закладывать при разработке систем (иначе откуда тогда столько систем, хранящих

пароль на доступ к базе данных в виде незашифрованного текстового файла?), так даже и те средства, которые в системах все-таки реализованы, при внедрении систем не настраиваются почти никогда (наверное, на это уже не хватает сил после борьбы с функциональностью). Например, из десятка компаний, в которых мне пришлось побывать за год, на половине администраторскими правами в приложении были наделены все пользователи, и практически на всех пользователей в базе данных присутствовали учетные записи с паролями «по умолчанию». И если настраивать межсетевые экраны уже многие научились, то добыча любой информации изнутри локальной сети – до сих пор задача, с которой может справиться любой, кто умеет давить на кнопки.

При этом компании достаточно серьезно говорят о конфиденциальности… Вон она, эта конфиденциальность, – висит в открытом доступе. – *Д. К.*

Права по разным ролям у одного пользователя должны объединяться, а не интерферировать способом, неизвестным даже самому разработчику. А это тоже бывает достаточно часто

Смежный вопрос, тоже связанный с защищенностью системы, – это протоколирование действий пользователя. *Любых действий и любого пользователя*. Почему-то разработчик, если его на это сподвигнуть, еще предусматривает фиксацию изменений, влияющих на финансовую отчетность (накладных, платежных поручений и кассовых ордеров), но традиционно плюет на протоколирование изменений сопутствующих справочников и, что еще страшнее, на протоколирование действий администратора системы.

А ведь с помощью таких дырок можно безнаказанно ломать и финансовые документы. Если некто хочет ликвидировать накладную о выдаче товара себе, любимому, или своему любимому контрагенту, можно не трогать саму накладную, а удалить контрагента этой накладной или заменить его на другого. То есть «всего-навсего» изменить справочник контрагентов. При отсутствии протоколирования действий администратора системы все можно провернуть еще проще: некто прописывает логин нового пользователя системы с правами изменять накладные, входит под этим логином в систему, удаляет накладную, потом снова заходит с правами администратора и удаляет «засвеченный» логин.

Грабли модные: xml внутри базы данных

При каждом скачке в развитии вычислительной техники у разработчиков информационных систем возникает эйфория, связанная со снятием ограничений на объемы хранимой информации и скорости ее переработки. Но всякий раз выясняется, что, даже вооружившись современными высокопроизводительными серверами, все можно спроектировать настолько плохо, что работать это не будет. Или будет работать годами.

Формат XML придумали для того, чтобы обмениваться информацией между разнородными системами, и в этом смысле его значение сложно переоценить. Но он хорош именно для обмена данными. Использовать этот формат для хранения больших объемов информации, предназначенной для постоянного использования, переработки и изменения, не следует. Для этого существуют базы данных и системы управления ими. Но вот незадача: чтобы манипулировать информацией в таблицах базы данных с разными полями, нужно писать разные процедуры, а в XML все можно делать ровно одной.

И обрадованный этим разработчик, «вооруженный передовой технологией», создает в базе таблицы с ключом и одним очень длинным текстовым полем, куда запихивает информацию в XML.

И все работает, пока такие записи нужны поштучно. Не так много времени требуется и для того, чтобы вывести пару десятков таких записей на экран. Но вот бизнес-заказчик просит отчет, который требует перелопачивания всех таких записей в базе. Процедуру для отчета написать получается, но работает она уже часами. Оно и понятно: все средства СУБД, заточенные для выбора нужной информации (например, индексирование), теперь применить нельзя, ибо нужно залезать в каждую запись, расшифровывать нотацию XML и только затем выяснять, нужна ли она для обработки.

Нетленные универсальные грабли

Еще одни вечные грабли – попытка вместо решения конкретной задачи создать универсальное решение. Эти грабли бывают программистскими (сооружается, например конструктор форм или генератор отчетов) или консультантскими (сооружается «универсальный модуль управления процессами» или еще какой-нибудь «модуль

управления понятиями»). Результат можно наблюдать во многих тиражных системах – каждая содержит по четыре-пять различных генераторов отчетов (причем на практике все равно отчеты либо программируются, либо получаются специализированными средствами напрямую из базы данных) и «модуль управления бизнесом» (при ближайшем рассмотрении – еще один генератор отчетов). – *Д. К.*

При следующей просьбе бизнес-заказчика обработка информации в отчете усложняется. Процедура снова пишется, но через полчаса после ее запуска сервер приложений падает, всхлипнув напоследок: «Out of memory»

Грабли детские: использование excel для обмена информацией

Описываемые грабли имеют небольшой размер, поэтому бьют не по лбу, а гораздо ниже. Но бьют гораздо чаще остальных. Это единственные грабли, на которые при мне наступили более ста раз.

Для работы с таблицами Excel штука очень удобная, но, к сожалению, обладающая зачатками интеллекта, который иногда применяет чрезвычайно не к месту. Например, если вы заранее явно не указали формат ячейки, в которую помещаете информацию, Excel сам догадывается, что вы имели в виду.

Рисуете вы накладную, записываете в ней цену 1 рубль 5 копеек, то есть 1,05, а Excel сам догадывается, что вы имели в виду 1 мая, что и записывает в ячейку. Вы ему кричите: «Я хотел ввести число!» – и он соглашается: «Число, так число», – и вы с изумлением обнаруживаете в ячейке 39569.00…

Пока вы вводили данные руками сами, все было не очень страшно: попили валокордина, установили у ячейки правильный формат и ввели то, что хотели.

Гораздо веселее, когда в Excel выплюнула отчет информационная система, а программист, который писал этот отчет, не знал, что надо описать форматы ячеек, в которые выводил информацию, или поленился это сделать. Потому что теперь мест в таблице, где стоит то 3 марта, то 31 декабря, может оказаться несколько тысяч.

Еще интереснее получается, когда вы пытаетесь с помощью таблиц в Excel выравнивать информацию в двух информационных системах: количество моментов, в которые этот технический гений сможет вам услужить, будет гораздо больше. И уже

несколько программистов должны будут не забыть предварительно описать форматы.

Как следствие, меняться табличной информацией лучше все-таки в старом добром формате DBF или в новомодном XML, которые даже Excel будет понимать правильно.

Шизофреническая глава про абсолютно нормальных людей

Неприличное слово из четырех букв: юзер

Я до сих пор верю в людей. Но не следует распространять эту веру на пользователей информационной системы. И все эти нравоучительные «Поставь себя на его место» здесь совершенно не годятся. Я на месте любого юзера все равно буду понимать, что я хочу сделать, оценивать последствия своего поступка, контролировать свои действия. Наверное, каждый айтишник участвовал в таком диалоге:

– Андрей, у меня компьютер завис.

– А на какие кнопки ты нажимала?

– Откуда я знаю?

Сколько предупреждений и вопросов ни задавай перед выполнением критической для системы операции, как ни заставляй пользователя выбирать ответ не по умолчанию, отвечать сначала «Да», потом «Нет», он все равно сначала произведет все необходимые манипуляции и, только увидев сообщение «Все проводки за 2005 год успешно удалены», воскликнет: «Ах, что же я наделал!» Поэтому обычному пользователю вообще нельзя доверять выполнение критических операций. Их должен производить адекватный администратор информационной системы по письменной заявке руководителя соответствующего подразделения.

> Кстати, это еще одна причина для грамотного разграничения доступа. Если мы не прячем информацию, это еще не означает, что мы готовы ее потерять. – *Д. К.*

Законы Мерфи никто не отменял, и если для уничтожения базы данных нужно нажать в правильной последовательности на двадцать семь клавиш, то клавиши будут нажаты.

В ИТ-подразделениях, которыми я руководил, с определенного момента неукоснительно соблюдалось правило: не проводить никаких критических операций в одиночку.

За компьютером в таких ситуациях обязательно сидят двое: один медленно набирает необходимые команды, а второй хватает его за руку, если, на его взгляд, команда сделает что-то не то. И

перед нажатием на клавишу Enter или кликаньем в кнопку «Пуск» обязательно выдерживается пауза, чтобы оба могли прочесть набранный текст.

Но попробуй добиться такой аккуратности от пользователя.

– А-а-а-а! У вас тут все так по-дурацки расположено, что я все время промахиваюсь мышкой!

– Так вроде между кнопками, которые вы перепутали, на вашем экране целый сантиметр.

– Все равно промахиваюсь! Вы хоть поставьте здесь запрос на подтверждение действия.

– Но ведь это замедлит вашу работу…

– Пусть замедлит, но зато я не буду ошибаться.

Мы добавляем вызов по нажатию кнопки модального окна с просьбой подтвердить действие.

– А-а-а-а! Как неудобно работать! Что вы заставляете меня подтверждать каждый чих? Я что, дура?

Один менеджер в течение полугода требовал от меня выводить в экселевом отчете по три числа в одну ячейку, потому что так удобнее их взглядом охватить. Я пытался объяснить, что отчет выводят в Excel не только для охватываний взглядом, но и для дальнейшей возможности обработки, а три числа в одной клетке обрабатывать средствами Excel затруднительно.

Тогда он хватал калькулятор с криком:

– Давай посмотрим, в каком из вариантов будет проще обработать!

– В том варианте, который я предлагаю, не нужен калькулятор. Я в том же Excel нажму значок суммы и выделю мышкой нужный мне столбец.

– А мне так неудобно!

Зачем нужны пользователи

– Зачем это я пойду к ненормальным? – пролепетала Алиса. – Я ж… Я лучше к ним не пойду…

– Видишь ли, этого все равно не избежать, – сказал Кот, – ведь мы тут все ненормальные. Я ненормальный. Ты ненормальная.

– А почему вы знаете, что я ненормальная? – спросила Алиса.

– Потому что ты тут, – просто сказал Кот. – Иначе бы ты сюда не попала.

(Льюис Кэрролл «Алиса в Стране чудес», пересказ Бориса Заходера)

Эти слова были впервые опубликованы в 1865 году. Но они про нас с тобой, брат-айтишник. Про нас, а не про пользователей.

Это пользователи – нормальные люди, а мы зачастую – нет.

Да, пользователи бывают и глупыми, и необразованными, но ведь и программисты тоже.

Именно пользователи создали этот мир. Это они кормят тебя, поят и одевают. Это они бороздят океаны и выходят в открытый космос.

Среди пользователей встречаются и умные, и красивые. Женимся мы обычно на пользователях и детей заводим с ними.

Если ты работаешь в производственной или торговой компании, то ведь именно пользователи в ней изготавливают товары и продают их. А мы – обслуга. Образованная, высококвалифицированная, но – обслуга.

И даже если ты работаешь в айтишной фирме, то деньги в ней делаешь не ты, а сейл. Ты можешь считать его тупицей, но все равно не получишь зарплаты, пока он не добьется нужных закорючек на нужных документах.

Поэтому радуйся тому, что у тебя интересная и интеллектуальная работа, но всегда помни, что без пользователей ее бы не было.

Мы – обслуга, и должны сделать работу пользователей более эффек тивной и удобной. Причем по возможности делать это незаметно, как хороший официант или уборщик, а не пытаясь выставить себя пупом вселенной.

Поражает, насколько большая часть айтишников функционально не в состоянии понять, что такое технология и удобство работы пользователя. Как будто за продумывание технологий и написание программ в человеческом мозгу отвечает ровно один участок, который, если он вообще есть, чаще всего специализируется ровно одним из описанных способов. А люди, которые и программы пишут, и понимают, как удобно работать пользователю, встречаются не чаще, чем шестипалые.

В одной из систем, которую я сопровождал, специальное подразделение занималось обработкой заявок других пользователей. Заявки выводились на экран в виде строк, общее количество которых за день доходило до тысячи. Когда другие пользователи завершали предварительную обработку заявки, она появлялась на том же экране при очередном обновлении. Но не обязательно в нижней части списка, а в произвольном месте.

За заявки, не обработанные в срок, сотрудников этого подразделения ругали и штрафовали. В ответ поступали жалобы, что уследить за всеми заявками невозможно.

Так система эксплуатировалась два года. *И почему-то никто не мог догадаться вывести на экран дату и время обработки заявки этим подразделением, чтобы таблицу можно было отсортировать по этому полю, или хотя бы галочку, что заявка обработана…*

Ну что здесь можно комментировать?

Зато сколько гонора обычно у айтишника, общающегося с пользователем! Причем чем меньше этот айтишник в состоянии понять, что и зачем просит пользователь, тем гонора больше. Апофеозом обычно является фраза «Я лучше вас знаю, что вам нужно!».

* * *

Мы сколько угодно можем плакаться друг другу на глупых и безграмотных пользователей, но они являются составной частью внешнего мира, в который мы интегрируем систему, и рассчитывать на замену этих пользователей на пользователей умных и грамотных можно с тем же успехом, что и на замену налогового законодательства в целях удобства программистов. А посему система ваша должна быть пользователеустойчивой и учитывать не только эргономику, но и психологию тех, кто систему использует.

В начале 1990-х, во времена перестройки, кодировки КОИ-7 и клавиатур ЙЦУКЕН/JCUKEN, мне попалось пособие для сотрудников МПС по ликвидации компьютерной безграмотности, в котором я обнаружил поразившую меня строку: «Для выхода из программы нажмите ЯУИТ». Посмеялся, конечно, а при встрече с автором этого произведения не преминул высказать ему свое недоумение. А автор ответил, что поставил в этом месте ЯУИТ вместо QUIT, равно как и другие кириллические заклинания вместо осмысленных английских слов, абсолютно осознанно: «Программа читает только 7 бит из 8, поэтому ей все равно, кириллицей или латынью введена команда. А для пользователей не все равно, они министерские работники, а это значит, что исполнительская дисциплина у них высокая и что никакой чушью их не удивишь. Если написано ЯУИТ, то и наберут на клавиатуре ЯУИТ, а вот английское слово приводит их в ступор, из которого они не

выводятся».

Крыть было нечем: автор был абсолютно прав.

Интерфейс пользователя: что такое хорошо

Понимание, что пользователь, даже когда мыслит, делает это совсем не так, как программист, не должно вас оставлять при разработке интерфейсной части системы. К примеру, нужно понимать, что изменение порядка следования пунктов главного меню, которое программер произвел, случайно зацепив локтем клавиатуру, пользователем воспринимается как вселенская катастрофа, практически конец света, хотя программисту для возврата прежнего порядка нужно нажать на пяток клавиш.

Понятия красоты и удобства у программиста и у пользователя тоже сильно разнятся. Частично это объясняется и тем, что пользователь проводит за экраном системы гораздо больше времени, чем программист, да и средний юзерский возраст сильно превосходит средний программерский. Вот традиционные нестыковки.

– Раскраска экрана в цвета попугая нравится не всем. Не следует ее применять для экранных форм, которые стоят перед глазами пользователя весь день. Если ваша тяга к выводу на экран всех цветов радуги непреодолима, то дайте пользователю возможность настроить цвета экрана.

– Я понимаю, что вы считаете всю информацию, доводимую до пользователя, важной. Но из этого не следует, что все сообщения нужно выдавать прописными буквами на красном фоне.

– Вы, может быть, удивитесь, но отнюдь не все жаждут видеть на экране максимальное количество информации. Очень многие хотят видеть только необходимый минимум, но в хорошо читаемом виде.

> Наверное, основное правило при проектировании интерфейса – это его простота. Если в глазах рябит от кнопок, полей, менюшек – скорее всего, интерфейс спроектирован плохо. – *Д. К.*

– У пользователей бывает разное зрение. У очень многих оно плохое.

– Следствие двух предыдущих пунктов: интерфейсы нужно тестировать не только на максимальном разрешении экрана, но и в режимах крупных букв и крупных значков. Почему-то в этих режимах у очень многих систем исчезает курсор, склеиваются или

налезают друг на друга поля и происходят другие мелкие пакости.

Иногда кажется, что разработчик в качестве основной поставил перед собой именно цель доведения пользователя до истерики.

Наиболее традиционным способом достижения этой цели является выдача сообщения «У вас нет прав на создание (изменение) этой записи» в тот момент, когда пользователь уже ввел не менее трех экранов информации. Другой способ, пользующийся большой популярностью, состоит в выводе курсора в поле, изменение которого пользователю запрещено, сразу при открытии экранной формы. Дальше пользователь или давит на кнопки, но ничего не происходит, или вводит/правит информацию, после чего система сообщает, что делать этого нельзя.

Еще очень забавно наблюдать за пользователем, которому модальное окно вывели ПОД окно, в котором он работает. Причем наиболее выдающиеся специалисты исхитрялись делать это еще в досовском алфавитно-цифровом интерфейсе.

Один раз в своей жизни я столкнулся с системой, в которой, захватив скроллбар (бегунок, полосу прокрутки) мышкой, можно было прокручивать информацию на экране, но как только ты скроллбар отпускал, экран перематывался обратно в исходное состояние. Слава богу, столь «выдающиеся» разработчики встречаются очень редко.

Есть немало навязчивых программистских «услуг», которые обычно совсем не нужны пользователю. Один из классических примеров – это сортировка в большой таблице, которую редактирует пользователь, производимая не по нажатию какой-нибудь кнопки, а в момент обнаружения системой измененной информации. Программеру кажется, что он сделал все правильно и оперативно, а бедняга пользователь никак не может понять, куда делась с экрана строчка, в которой он только что был (а она убежала упорядочиваться). Еще более сильный эффект достигается, когда пользователь заменяет блок информации, захватывающий несколько соседних строк: после выполнения команды «Вставить» соседние строки разбегаются по таблице, как тараканы.

К той же категории шуток относится выполнение отчета, имеющего параметры, ровно в тот момент, когда пользователь выбрал его в меню. Поскольку в этот момент никакие параметры еще не введены, отчет исполняется с параметрами по умолчанию, и через полчаса, в которые работать не может не только этот

пользователь, но и еще сотня других, подключенных к системе, на беднягу изливается несколько тысяч строк, хотя нужна ему была только одна строка, а расчет с нужными параметрами должен был занять доли секунды.

Постоянные конфликты, возникающие из-за различия менталитета айтишников и юзеров, потребовали озвучить еще одно общее правило: **если вам кажется, что прибамбас, о котором вас никто не просил, окажется для пользователей системы очень полезным и удобным, то добавьте его так, чтобы он не проявлялся при установке рабочего места с параметрами по умолчанию. В крайнем случае, так, чтобы его можно было при желании отключить.**

Финишная кривая

Что такое «программа заработала»: когда начинать внедрение?

Давным-давно, когда я работал преподавателем, один из моих студентов отчитался по результатам своей работы так: «Программа уже работает, но пока дает неверные результаты». С тех пор эта фраза вошла в классику программистского фольклора.

Однако по сей день однозначного ответа на вопрос, что же такое работающая программа, на мой взгляд, не получено и вряд ли когда-нибудь будет получено. То, что во всех больших программах всегда содержатся ошибки, общеизвестно. Хотя неизвестно, что такое ошибка (этот вопрос долго и нудно приходится обсуждать с разработчиком при каждом составлении договора, а потом не менее долго и не менее нудно доказывать ему, что предъявленные результаты работы системы говорят об ошибке, и в тысячный раз выслушивать бред типа «Это не ошибка, раз вы не можете ее повторить…»).

На эту тему написаны сотни умных книг и тысячи еще более умных статей, в том числе с применением теорий вероятностей и массового обслуживания (вывод моей студентки, воспользовавшейся одной такой книгой, я прочитал в ее дипломном проекте: «Таким образом, еще через два часа эксплуатации в программе будет выявлена последняя ошибка»).

Не залезая в очередной раз в теоретические дебри, берусь утверждать, что в вашей системе ошибки есть сейчас и будут всегда. Хотя бы потому, что в качестве операционной системы хотя бы некоторых рабочих мест у вас стоит небезызвестный Виндоуз небезызвестной мелко-мягкой корпорации.

Но, даже зная это, вы должны решить, когда же можно начинать внедрение. И ваше руководство совершенно не будет волновать, что этот вопрос теоретически неразрешим.

Единственный ответ, который я знаю: внедрение надо начинать в тот момент, когда у вас появится уверенность, что *это* заработает (хотя бы основные функции). Если такая уверенность не появляется слишком долго, надо сменить базовый софт, бригаду программистов или профессию. Впрочем, о последнем позаботится ваше начальство.

Итак, равнение на Гейтса, и – вперед!

Здесь можно дать несколько советов, позволяющих

если не провести внедрение безболезненно, то хотя бы минимизировать его разрушительную силу.

• Всегда разбивайте работы на небольшие фрагменты, результат которых можно увидеть. Иначе мы уподобимся менеджерам проекта, которые, глядя на график работ, неизменно видят, что работы «выполнены на 80 %» (они всегда будут выполнены на 80 %, пока за день до сдачи системы не выяснится, что работы… выполнены на 80 %).

• Внедрять лучше небольшие функциональные модули, поэтапно. Ввод всей системы в один момент недаром именуется в западной литературе методом «большого взрыва». Плюсы поэтапного внедрения очевидны – с одной стороны, вы ограничиваете масштаб возможного бедствия, с другой – сразу получаете некий результат. Не стоит забывать о том, что вероятность успеха внедрения обратно пропорциональна сложности системы – поэтапный ввод позволяет эту сложность сократить.

• Всегда определяйте реальные приоритеты реализации функциональности. Да, это будет непросто, и придется много додумывать самому (если это отдать на откуп пользователю, то наивысший приоритет приобретет любая ерунда). Зато, если это получится, при неблагоприятных обстоятельствах пользователь будет просто недоволен, но при этом предприятие не прекратит свое функционирование.

• Всегда выбирайте для старта развертывания системы момент с наименьшей загрузкой (чаще всего таковой имеется). Сбой системы, обеспечивающей процесс продаж под Новый год, – совсем не то же самое, чем сбой системы в октябре (разумеется, в каждой компании свои периоды пиковой загрузки).

Предусматривайте резервные пути жизни компании в случае сбоя. Понятно, что построение таких мостов – это дополнительное время, но если «ой» произойдет, время все равно будет затрачено (только работать в этом случае придется почему-то и днем, и ночью, да еще вдобавок компания может понести значительные потери). Процесс возведения резервных механизмов и временных процессов (а иногда и систем) называется управлением рисками, и о нем много любят говорить, но на практике предпочитают почему-то бороться с проблемами, а не предотвращать их. – *Д. К.*

Гендерные аспекты внедрения

Я здесь не совсем про то, о чем вы подумали. Интеллект по людям распределяли, конечно, неравномерно, но совсем не по гендерному признаку. И женщин, понимавших и решавших задачи своей компании гораздо лучше мужчин, я встречал немало.

Тем не менее женщины от мужчин все-таки отличаются. Например, у них гораздо лучше обоняние. Поэтому, чтобы внедрить что-либо в женских коллективах, вы должны обеспечить как минимум ежедневную чистку зубов, мытье под душем, смену носков и трусов всеми сотрудниками, которые обследуют и обслуживают соответствующие подразделения. Нужно также провести разъяснительную работу и с мужчинами, обливающимися пахучими лосьонами, поскольку с такими тоже очень трудно находиться в одном помещении долго.

Если вам не удалось сделать ваших внедренцев невонючими, не удивляйтесь, когда услышите, что внедряемое решение работает плохо, медленно и неправильно, а ваши сотрудники не в состоянии внятно объяснить, как всем этим пользоваться. Впрочем, последнее утверждение верно во всех случаях: **воняющий человек может донести до окружающих ровно один месседж**.

Ввод в эксплуатацию

Я умышленно не написал «опытную», потому как давно перестал понимать, что это такое. Этот термин, может, и был хорош – но только для внедрения автоматизированной системы «с чистого листа», взамен системы управления с бумажными документами и способами расчетов на калькуляторах или счетах, что сейчас можно встретить разве что в небольшом магазинчике, но уж никак не в крупной фирме.

Опытная эксплуатация системы в этом случае подразумевала сохранение бумажного документооборота и хранение документов на бумажных носителях. А что имеется в виду под опытной эксплуатацией, если одна автоматизированная система заменяет другую? Параллельная эксплуатация двух разных автоматизированных систем?

До первой попытки я тоже считал, что это можно и нужно сделать. Но при внимательном изучении вопроса выяснилось, что:

1. Вычислительные мощности для обслуживания сразу двух систем нужно увеличить вдвое, а обслуживающий персонал – чуть

ли не в три раза. После опытной эксплуатации лишний персонал нужно уволить так, чтобы у вас сохранилась информация хотя бы в одной из информационных систем. Что делать с лишней вычислительной техникой, мне совсем не понятно;

2. Результаты работы систем практически невозможно сравнить:

– в новой системе используется другой справочник товаров, а в результате инвентаризации, с которой начинается внедрение, были ликвидированы последствия пересортицы: некоторые товары, фигурировавшие в старой системе под разными названиями, признаны одним и тем же, а другие товарные позиции разделены на несколько;

– в старой системе товар при резервировании списывался со складского остатка, а в новой он остается на складе, но не может быть отпущен;

– в старой системе учет велся по товарам по средневзвешенной цене, а в новой учет ведется по партиям, на каждую из которых фиксируется своя цена;

– в старой системе услуги (например, доставка товара) включались в приходные и расходные накладные как товарные позиции, и теперь их на складе 2744 штуки;

– для управленческого учета используются другой план счетов и другая аналитика.

Что касается применения понятия «опытная эксплуатация» как критерия отлаженности технологии и отсутствия ошибок в программах, то в давно работающей системе обычно технология более отлажена и ошибок меньше. И то не всегда, потому что и технология, и программное обеспечение модифицируются постоянно.

> Критерий, отличающий опытную эксплуатацию от рабочей, все-таки есть. Если ответом на вопрос обеспечения непрерывности работы системы является регулярное резервное копирование, то система в рабочей эксплуатации. Если ответ сложнее (параллельная эксплуатация, ручная работа и т. п.), то система еще в опытной. Главное, не забыть о том, что ответ на такой вопрос должен быть всегда. – *Д. К.*

Я могу предложить другие признаки, по которым можно определить успешность хода внедрения:

– вы снова ночуете дома;

– вам удалось увидеть своих детей не спящими;

– с задачи сняли дополнительный персонал, или сотрудники

отдела автоматизации заменены штатными операторами;

– руководство сообщило вам, что больше не нуждается в сотруднике такой высокой квалификации, как ваша.

Разумеется, перечисленные события должны быть связаны непосредственно с внедрением, а не с тем, что вы помирились с женой или поссорились с начальством.

К внедрению нельзя приступать с некомплектным штатом, плохим состоянием здоровья, в состоянии стресса, не связанного с работой.

Перед началом внедрения вы должны быть убеждены, что не только вы, но и ваше начальство и коллектив, с которым вы собираетесь работать, отчетливо понимают, что неправильные решения на этом этапе могут превратить процесс в бесконечный.

Пример. Пересчитав остатки на складе, вы начинаете вводить их в систему. Склад в это время продолжает работать: товары привозят и развозят по торговым точкам. В результате к моменту окончания ввода остатков состояние склада в системе уже отстает по времени от текущего состояния склада, и вам нужно ввести в систему ворох накладных, накопившихся с момента снятия остатков. Теперь, если число работающих операторов больше необходимого для штатной работы системы, вы сможете через некоторое время сократить отставание и привести систему к реальному времени. Если у вас только штатное количество операторов, отставание в лучшем случае сохранится постоянным, а если меньше, то будет расти бесконечно.

Учтите при этом, что скорость ввода информации при смене системы падает из-за изменения технологии, форматов данных, кодов и наименований.

Понятно, что набор дополнительного персонала никого не радует: мало того, что приходится платить дополнительные деньги, еще и неясно, куда девать этот персонал после окончания внедрения. Но нужно трезво относиться к обычному в таких случаях предложению сверхурочно поработать штатным оператором, пусть даже и за дополнительную оплату. При увеличении продолжительности рабочего дня производительность персонала падает, а количество ошибок возрастает так, что иногда бывает проще очистить базу данных и снова начать все с самого начала, чем найти и исправить все эти ошибки.

И руководители всех звеньев, и рядовые сотрудники должны ясно представлять причины, приведшие к необходимости замены одной автоматизированной системы на другую. Первое время тяжелее работать станет всем, а потом (да и то лишь в лучшем случае) выгоды от использования системы получат только некоторые.

К некоторым почти всегда относится руководство, которое станет получать более точную и оперативную информацию, и почти никогда не относятся операторы, которым придется переучиваться, привыкать к новым правилам ввода информации (может быть, не худшим, но *другим*, что раздражает неимоверно), а потом еще и обрабатывать большие, чем раньше, объемы информации.

Желание операторов вернуться к старой системе может приводить к саботажу новой и, в исключительных случаях, к диверсиям против нее. Мягкие методы противодействия саботажу не всегда дают эффект, и иногда приходится идти на полную замену операторского персонала.

Есть еще одно очевидное требование, выполнения которого не так-то просто добиться в наше время. Как минимум за две недели до начала внедрения (большего срока вам все равно не дадут) надо наложить мораторий на все изменения в технологии работы и на требования срочно разработать дополнительный отчет или печатную форму.

К сожалению, кроме субъективных причин типа хозяев и начальников «с шилом в одном месте» («Я не могу ждать! Это приносит мне ежедневные убытки в пять тысяч долларов!» – про отказ срочно изменить способ сортировки товаров в отчете), существуют еще и объективные (введен или отменен очередной налог; Мосалкогольконтроль в пятый раз изменил форматы данных, которые ему нужно передавать еженедельно; сделалось 17 августа 1998 года, и остатки склада в рублевых закупочных ценах превратились в полную абстракцию, не связанную с реальной жизнью, а формирование цен реализации как функции от закупочных стало верным способом разорения).

Не со всем из перечисленного можно бороться, но еще один совет я все-таки могу дать: не привязывайте начало внедрения к отчетным датам (начало года, квартала, месяца), потому что моменты введения в действие законов и постановлений тоже привязывают к ним. Исключением из этого правила служит подсистема официальной бухгалтерии, которую иначе как с начала квартала ввести в эксплуатацию не получается. Но это является и дополнительным аргументом для того, чтобы начинать эксплуатацию складских подсистем в другое время.

Место для подвига

В нашей стране понятие подвига не вполне однозначно. Даже в детстве меня удивляло, что подвигами объявлялись ситуации, в которых человек погибал, а задание его оставалось невыполненным.

Военный подвиг, конечно же, всегда связан с риском гибели. Но если двое сделали что-то выдающееся, при этом один из них погиб, а второй – нет, то славы достойны оба, а в пример лучше ставить второго. Камикадзе все-таки не слишком эффективны.

В нашей отрасли, конечно, жизнью обычно не рискуют, только здоровьем. Это попадает под стандартное определение подвига трудового. И в некоторых случаях без подвига в нашем деле не обойдешься.

Восстановление информационной системы, обеспечивающей предприятия с круглосуточным циклом работы, должно производиться ровно с того момента, как она обрушилась, независимо от того, произошло это днем или ночью, и продолжаться до ее восстановления, полного или частичного. Я даже не говорю про системы, обслуживающие энергетику или банк крови. И компанию – дистрибьютора скоропортящихся продуктов или ежедневных газет сейчас можно легко разорить, отключив от информационной системы на несколько торговых дней. Понятно, что лучше от таких ситуаций защищаться, но иногда они все-таки происходят.

Бывают и плановые подвиги. К примеру, система у вас работает круглосуточно и ежедневно, а вам нужно внести в нее существенные изменения. Остановка функционирования будет, конечно, запланирована заранее, но работы придется проводить, пока они не закончатся, а это может растянуться на несколько суток. А число сотрудников, способных такие изменения произвести, не убив систему окончательно, всегда минимально. Никто не будет держать их на работе ровно для таких случаев. Мне и штатным-то количеством персонала, оговоренным с руководством, последние десять лет укомплектовать свои подразделения не удавалось ни разу.

В перечисленных случаях приходится работать по несколько суток подряд, иногда отключаясь непосредственно на рабочем месте на время длительных операций, оставив у монитора оператора, готового разбудить тебя словами «оно закончилось» или «оно сломалось».

Никуда не деться и от подвигов помельче. Если со складов в магазины товар развозится по ночам, то вас иногда будут будить ночью по производственным вопросам. Если в выходные магазины

компании продают товара в три-четыре раза больше, чем в будни, то не удивительно, что к вам будут приставать и по выходным.

Но подвиги в ИТ распространены гораздо больше, чем это необходимо. Связано это как с менталитетом самих айтишников, все еще жаждущих романтики от своей работы, так и с менталитетом хозяев и начальников, очень благосклонно относящихся к жертвам в собственную пользу. Но есть ли от этого польза на самом деле?

Много ли вы знаете программистов, способных эффективно работать более двенадцати часов подряд? Я лично знал ровно одного, который переставал делать в коде даже синтаксические ошибки на 26-м часу непрерывной работы. Но, когда эта работа все-таки заканчивалась, он сначала отсыпался, а потом уходил в недельный запой.

Так что надежда на то, что проект можно закончить раньше, если программисты будут работать больше, не имеет под собой никаких оснований, как бы ни хотелось на это надеяться всякий раз, когда сроки проекта срываются. Обычно происходит ровно противоположное: перестав соображать, ваша бригада программистов и настройщиков вставляет в код и настройки такое количество ошибок, что их потом приходится отлавливать месяцами.

Работу программиста по сдвинутому графику, когда он приходит на работу после обеда, а уходит заполночь, я вообще не расцениваю как подвиг. В большинстве случаев это обычная расхлябанность, в остальных, возможно, попытка поймать свой пик суточной активности, поэтому я стараюсь таких вещей не запрещать, но и ни в коем случае не стимулировать.

> В одной из торговых фирм, где я айтидиректорствовал, работал программист, совершавший подвиги постоянно и требовавший за это искреннего почтения окружающих, высокой зарплаты и премий за сверхурочную работу. То есть программистов было несколько, но подвиги совершал только один. В какой-то момент до меня дошло, что подвиги его чрезвычайно однообразны: сначала в обстановке совершенно некритической писался код, который потом внедрялся в действующую систему. Поскольку код содержал ошибки, в системе начинались сбои. И тогда, уже в аварийной ситуации, ночью или в воскресенье, появлялся наш герой, чтобы закрыть амбразуру своей грудью. Стоп. Конечно, замечательно, что он детей из пожара выносил, но ведь перед этим он сам дом и

поджигал. И я добился увольнения этого человека. Подвиги исчезли вместе с ошибками в коде.

Кто виноват?

Я уже забыл, в какой книге в 1970-е годы я прочитал про этапы разработки любых больших систем, но на всю жизнь запомнил, что последние три этапа – это:

– поиски виновных;

– наказание невиновных;

– награждение непричастных.

Книга была американская, что доказывает, что в основных реакциях все люди одинаковы. Последний этап иногда отсутствует (особенно если нет денег), зато два предыдущих следуют за окончанием разработки или опытной эксплуатации, как осень после лета.

Итак, не думайте, что вам за вашу работу ничего не будет. Будет, и еще как.

Когда меня отчитывали за то, что во всех магазинах нашей корпорации, независимо от местоположения, цены одинаковых товаров одинаковы, что противоречит азам капиталистической торговли, я обиделся и потребовал у руководства конкретизировать мою личную вину:

– Программное обеспечение готово, чтобы вести разные цены?

– Ну, готово.

– Я говорил, что, по моему мнению, цены должны быть разными?

– Ну, говорил.

– Так в чем же я виноват?

– А почему ты нас не убедил, что это правильно?!!

Впрочем, это самый легкий случай. Мне известен разработчик программы учета, к которому клиент привел своих бандитов с объяснением, что он своих долгов отдать не может, потому что не может понять, кто должен ему самому, и все из-за этой дурной программы. К счастью интеллекта бандитов оказалось достаточно для правильного разрешения конфликта.

Заключение

После прочтения этой работы может сложиться впечатление, что внедрить автоматизированную систему вообще невозможно.

Практически на каждом этапе каждого внедрения такое же впечатление складывается и у меня, так что мне приходится напоминать себе, что у меня за плечами есть опыт успешного внедрения систем, да еще и немалый.

Я должен также признаться, что по результатам внедрения мне несколько раз платили обещанную заранее премию.

Это должно вернуть вам утраченный оптимизм.

Так что успехов вам.

Приложения

Приложение 1. Разбор задачи для собеседования с программистами

Итак, разбор задачи.

Тест проверяет достижения соискателя скорее по п. 2 перечня требований, приведенного перед задачей, чем по п. 3 (если вы, конечно, еще помните, что там написано) Одновременно вы получаете представление об аккуратности кода и «доверчивости» при получении исходных данных. Вот решение, которое я хотел увидеть, без заморочек синтаксисом языка.

Комментарий: если поезда могут приходить раньше или опаздывать более чем на половину суток, то для решения задачи необходимы дата фактического прибытия и прибытия по расписанию. Далее задача решается в предположении, что время опережения и опоздания не превосходит 12 часов.

```
ЕСЛИ НЕ КОНТРОЛЬ_НА_ВРЕМЯ (ВРЕМЯ_РАСП) ИЛИ
НЕ КОНТРОЛЬ_НА_ВРЕМЯ (ВРЕМЯ_ФАКТ)
ТО ВЫЙТИ.
```

Комментарий: Я бы удовлетворился и без описания функции КОНТРОЛЬ_НА_ВРЕМЯ.

```
РАЗНОСТЬ:= МИНУТЫ (ВРЕМЯ_РАСП) – МИНУТЫ (ВРЕМЯ_ФАКТ);
```

Комментарий: 1440 = 24 х 60 – количество минут в сутках, а 720 – в полусутках.

```
ЕСЛИ РАЗНОСТЬ < = —720
ТО РАЗНОСТЬ:= РАЗНОСТЬ + 1440
ИНАЧЕ ЕСЛИ РАЗНОСТЬ > = 720
ТО РАЗНОСТЬ:= РАЗНОСТЬ – 1440;
ЕСЛИ РАЗНОСТЬ = 0
ТО ПЕЧАТАТЬ («ПРИШЕЛ ВОВРЕМЯ»)
ИНАЧЕ ЕСЛИ РАЗНОСТЬ > 0
ТО ВЫВЕСТИ («ПРИШЕЛ РАНЬШЕ НА», ЧАС_МИН (РАЗНОСТЬ))
ИНАЧЕ ЕСЛИ РАЗНОСТЬ < 0
ТО ВЫВЕСТИ («ОПОЗДАЛ НА», ЧАС_МИН (— РАЗНОСТЬ))
```

Если испытуемый не заметил, что поезд, прибывающий по расписанию в 0.05, а фактически прибывший в 23.55, не опоздал на 23 часа 50 минут, а приехал раньше на 10 минут, то это очень грустно. Если увидел, но не сообщил вам (устно или в

комментарии к тексту), что решение не работает при расхождениях с расписанием более полусуток, то это просто грустно. Кодировать по блок-схемам последние сорок лет уже не требуется, а на что еще такой годится?

Наиболее поразившей меня за последнее время реакцией на эту задачу было удивленное «Как нет дат в исходных данных? Без дат вообще нельзя решить, без дат нет метода!». Я сразу почувствовал себя таким старым… Правда, как-то мы без методов обходились, поскольку объектно-ориентированных языков тогда еще не было.

Приложение 2. Из записных книжек периода внедрения

Обследование и разработка технического задания
Начал новую жизнь: стер куки.

* * *

Этап обследования предприятия: слепой объясняет глухому, как выглядит Земля из иллюминатора космического корабля.

* * *

У нас же госучреждение. Шаг вправо, шаг влево – сразу служебная записка.

* * *

Сложно будет объяснить разработчикам, что «отдел» и «Отдел» – это разные виды подразделений, находящиеся на разных уровнях структуры. «Отдельный Отдел», наверное, тоже писать не стоит. Придется каждый раз писать «отдел (в составе Управления)» и «Отдел (вне Управлений)».

* * *

Информация отдела кадров: «Руководство не возражает, чтобы начальство отпустило сегодня своих подчиненных в 17 часов». Надо привыкать к терминологии. Вчера, например, вызвал их искреннее возмущение неправильно заполненной ежедневной справкой: «Неужели вы не понимаете, что ваш Т-ов находится не в отпуске, а в отгуле в счет отпуска?»

* * *

– В каком порядке визируется договор?
– Договор визируется в хаотичном порядке.

* * *

Моя профессия уже давно стала бы мне невыносимо скучна, если бы я не менял предметные области. Ну где бы я еще узнал про бланковый индоссамент и нетелей средней степени стельности, да еще и в одном договоре?

* * *

На то, чтобы объяснить своим сотрудникам, что смешного в выражении «племенной боров», ушло полчаса.

* * *

Разбирали с программистом бухгалтерский сленг. Пришлось даже залезть в Даля, чтобы выяснить происхождение некоторых слов. Выяснилось, что «крыжить» происходит от «крыж», то есть крест, и исходно значило «помечать крестом», хотя сейчас большинство бухгалтеров крыжат галочками. Заодно разобрали слово «херить», происходящее от «хер», старого названия буквы Х, и означающее по Далю перечеркивать или помечать косым крестом. Программист подумал и сделал вывод, что херить – это крыжить под углом пи на 4.

> Особенно креативно называются ситуации, в которых одному объекту необходимо сопоставить некий другой (например, связать задолженности и оплату). Обычно это именуется спариванием, сращиванием (это бухгалтерский фольклор), применением (а вот это уже из языка переводчиков системы Navision. Так и написали в интерфейсе – «применить операции». Нет чтобы голову применить).
>
> Что до термина «крыжить», то по тому, понимают ли его с первого раза, можно безошибочно определить автоматизатора. – *Д.К.*

* * *

Любые обсуждения любых проблем с любым сотрудником X из множества {А, Б, В} на треть состоят из попыток доказать утверждение, что Y из множества {А, Б, В} – козел, где Y, как вы

догадываетесь, не равен X.

* * *

После обсуждения очередной задачи я обязательно оформляю задание на программирование в письменном виде и даю подписывать сотруднику, эту задачу поставившему. Менеджер А такие постановки читает внимательно и в процессе чтения крыжит, то есть помечает плюсиком каждый пункт, с которым согласился. В процессе согласования задания он пользуется почти такими же подсказками, как в игре «Кто хочет стать миллионером?»: «Помощь зала» (допрос подчиненных), «50 на 50» и «Звонок богу».

* * *

Счастливы те, кто понимает разницу между учетом и регистрацией

> Разница заключается в наличии проводки в главной книге. – *Д.К.*

* * *

Мне приходится ходить в бухгалтерию, потому что по телефону полную информацию от бухгалтеров получить нельзя: часть ее передается только с помощью жестов.*(Федор Клабуков.)*

* * *

Если в компании для документооборота используется электронная почта, никогда не задавайте руководству два вопроса в одном письме: если вы и получите ответ, то только на первый вопрос. Впрочем, в обычной служебной записке тоже не следует поднимать сразу две проблемы. Этот документ служит исключительно для наложения резолюции и направления по инстанциям (исполнять или согласовывать дальше), поэтому в нем глупо просить что-то у двух подразделений: документ окажется в итоге только в одном.

* * *

Инструкция по организации инструктажа по пожарной безопасности.

* * *

На станции Лосиноостровская кассир продает билеты как на обычные электрички, так и на скоростной поезд «Спутник». Для этой цели у него два одинаковых терминала с двумя принтерами для печати билетов и двумя программаторами для чиповых карт.

Специально сравнил билеты: ИНН на них стоит один и тот же. Лишние тысячи три долларов на каждое рабочее место вместо подумать. Ну действительно, зачем думать, если эти деньги можно собрать с пассажиров. Куда ж они денутся…

* * *

Наказывать за отсутствие мозгов можно только из зависти.

* * *

– Не отвлекайте меня вопросами, я же руковожу.

* * *

У этого алгоритма логика, конечно, развесистая, но простая. *(Андрей Янин.)*

* * *

Можно ли обозвать генерального директора словом «пользователь»? Или лучше выделить отдельный АРМ руководителя, пусть он даже от АРМа пользователя ничем не отличается?

И может ли гендиректор изменить или добавить запись в справочнике контрагентов?

А если не может, то кто ему об этом скажет? И нужно ли ему об этом вообще говорить?

* * *

В большинстве компаний, в которых мне приходилось работать, штат делится на три части: руководство, сотрудники и «тоже люди».

Наличие последних становится очевидным по употреблению фраз «Операторы тоже люди», «Продавцы тоже люди» etc.

«Программисты тоже люди» я тоже слышал.

* * *

При обсуждении некоторых вопросов появляется желание предложить каждому участнику совещания надеть на голову обруч. Чтобы ни у кого голова не треснула. Иногда и самому надеть хочется.

Я подозреваю, что короны в Европе именно таким способом появились. Сначала это был просто обруч, который надевал правитель, чтобы у него при решении сложных государственных задач не треснула голова. И только потом эти обручи стали делать из золота и украшать драгоценными камнями.

А сегодня до меня наконец дошло, почему к короне стали приделывать вертикальные зубцы, а потом и большое количество обручей. Похоже, что корона исполняла еще и функцию радиатора охлаждения для перегретой от размышлений головы короля или герцога.

Кстати, для теплоотвода идеальной была бы серебряная корона, затем следует корона из чистой меди (а вот бронзовая будет уже в четыре раза хуже) и только после них – золотая, а потом алюминиевая (но его еще не умели добывать). Применение других материалов было бы существенно хуже.

А Владимир Михалев подсказал, что процедура помазания на царство служила на самом деле для нанесения на голову будущего монарха термопасты, обеспечивающей высокий теплообмен между головой и короной.

А Intel все это по второму разу изобрел.

Доработка функционала

Смотрел настройки документов. Настройщик так и не понял, почему я веселюсь при виде экранных форм «Разнорядка» и «Страховой полюс».

* * *

– Этот модуль работает, зуб даю.

– Значит, у тебя 32 попытки.

* * *

Руководитель проекта от фирмы-разработчика грустно говорит:

– Кнопку для массового пересчета всех договоров вы уже заказали. Меня удивляет, что вы еще не заказали кнопку «Уволить всех».

* * *

Программист, отвечающий у разработчика за работу со словоформами, узнал, что в русском языке шесть падежей, а не семь, как он предполагал. Он мне объяснил, что перепутал с количеством цветов радуги. Хорошо, что число апостолов первого круга ему неизвестно.

* * *

– Получил письмо от вашего директора. Он пишет, что описанная проблема легко решается. Но если она легко решается, то почему она не решается?

– Ну как же… потому что она решается легко, но до-о-о-олго….

* * *

– Вова, у них что, в системе нет механизма транзакций?

– Есть у них в системе механизм транзакций. Просто они им не пользуются. И голова у них, наверное, тоже есть.

* * *

Софт, не отчуждаемый от разработчика. Никаким способом не отчуждаемый.

* * *

Внедряемая информационная система при заполнении приказа о приеме на работу нового сотрудника задает вопрос: «Создать новое физическое лицо?»

> Внедряемая система при попытке ввести код страны «Россия» в заказ выдает сообщение «Выбрана неправильная страна». – *Д.К.*

* * *

Внедренцы хотят получить дополнительные деньги. На вопрос «За какие работы?» отвечают: «За доводку системы на площадке заказчика». При этом деньги за доработку и настройку они уже взяли. Бурно не понимаю, какую такую доводку системы нужно проводить после ее доработки. Один из моих сотрудников меланхолично замечает: «Теперь систему нужно подогнать по месту с помощью напильника».

* * *

Становлюсь нарицательным. Передали разговор между директором и сотрудником фирмы-разработчика: «Что вы со мной разговариваете как Орлов!»

* * *

Обсуждали идею создания специального продмага для женатых программистов. В таком магазине обязательно должны продаваться продукты с названиями «Мясо хорошее», «Картошка крупная», «Бананы негнилые» и «Колбаса какая-нибудь». Мне сказали, что я забыл главные продукты: «Что-нибудь к чаю», «Что-нибудь на десерт», «Что-нибудь на ужин» и продукт-апофеоз «Что-нибудь из еды». Конечно, в этом магазине должны быть и соответствующие единицы измерения продуктов: «немного», «не очень много», «на один раз», «сколько я обычно покупаю».

Поехали

Проблемы с быстродействием. Внедренцы советуют поставить пользователям более мощные компьютеры. Удивленно сообщаю им, что в ТЗ их система определена как система с тонким клиентом. Похоже, в процессе доработок клиент не на шутку располнел.

* * *

На экране системы сопровождения программного продукта рядом выводятся два поля: «Заявитель» и «Плановая дата устранения».

* * *

Модальное окно «Вы действительно хотите выйти из программы?» должно содержать три варианта ответа: «Да», «Нет» и «Еще как».

* * *

Метод разработки: картошку порезать, пожарить, подать на стол, а если заказчик заметит, что картошка не чищена, начать чистить.

* * *

Сегодня сформулировал функции ключевых фигур в ИТ-проекте.

Таких фигур три: руководитель проекта, главный разработчик и главный аналитик.

Аналитик отвечает за вопрос «почему разработали?», разработчик – «как разработали?», а руководитель – «кого за это нужно взгреть?».

Как следствие, на вопрос «что сделали?» ответить не может никто.

* * *

А я понял, почему в этой версии системы не наблюдается никаких следов предыдущего опыта внедрения информационных систем: все обладающие памятью запоминают обещания Б. и увольняются, когда он их не выполняет. Остались только амнезированные.

* * *

Нет, смертная казнь для некоторых разработчиков – это слишком гуманно. Я бы приговаривал их к пожизненной эксплуатации собственных программных продуктов

Предлагаю пожизненное форматирование дискет. – *Д.К.*

* * *

Причина увольнения: саботировал невысказанное желание руководства.

* * *

Тоже стиль руководства: всегда принимать меры и никогда не принимать решений. И ведь он хорошо держится на своей должности.

* * *

Поскольку любая система работает без ошибок только в одном случае – когда ею не пользуются, – то для адекватного отражения реального мира в системе следует остановить реальный мир.

Похоже, что большинство IT-консалтеров именно это и пытается сделать.

* * *

Состояние внедрения: кома.

* * *

База загигела. Федя сказал, что размер базы превысил гигабайт, подчеркнуто вскользь, между прочим. Но я заметил. Все-таки событие. Нам еще долго предстоит выслушивать истерики пользователей и утирать их слезы, воевать с внедренцами по поводу каждой формы ввода, вытрясать недоделанные куски, но все-таки речь уже идет не об игрушке, а об информационной системе, в которую мы запихнули три года

работы нашей компании.

* * *

Уволен менеджер по сопутствующим товарам. Пока ехали из офиса на склад, я поинтересовался у исполнительного директора, что произошло.

– Я, по-твоему, не права?

– Права, конечно. Она милая женщина, но я бы ее уволил на год раньше, когда мы ей не смогли объяснить, что нельзя менять розничную цену, не проводя инвентаризации. Но сейчас какие события к этому привели?

– Как какие? – удивляется. – Вы же сами сделали новый отчет по товародвижению, который я просила.

– Но в отчете про уволить ничего не было.

– Зато там было про торговую наценку. А когда я увидела, что торговая наценка по всем товарам, отпущенным со склада за месяц, меньше фонда заработной платы склада, я уже сама догадалась, что делать.

Начинаю понимать, что на самом деле значит «использовать информационную систему».

Эксплуатация

В два часа ночи дома звонит телефон. Старший оператор интересуется, не звонил ли я ему сейчас.

* * *

Данные не обязательно копировать ежедневно. Достаточно это делать перед аварией с базой.

* * *

Узнал, что операторы за глаза зовут меня «папа».

Сначала расстроился. Потом понял, что это все-таки не худший вариант.

* * *

Напутствие гендиректора: «Делайте что хотите, работайте

ночами, но результат должен быть через час».

* * *

Не издеваясь и не повышая голоса, абсолютно спокойно объяснить мальчику из техподдержки, что «артикль» и «артикул» – это разные слова, сообщить их значения, а потом попросить писать слово «лучше» через «ч», а не через «д», а то мне неудобно показывать журнал заявок исполнительному директору.

* * *

Этот мальчик очень старательный. Поняв, что хромое правильнописание несколько меня нервирует, он теперь все тексты, которые я должен прочесть, проверяет с помощью Ворда. И Ворд сообщает ему, что в предложении «Заместа поля Версия мы можем писать имя компьютера» это самое «заместа» нужно писать раздельно.

* * *

Грустная шутка: «Компания „Супер-Софт“ признала неисправимой в текущей версии своей системы очередную орфографическую ошибку».

* * *

Мой сотрудник звонит пользователю и представляется: «Это Йохан-программист».

Он не кривит душой и не нарушает правила: футбольно-фанатичный папа назвал его в честь Йохана Круиффа, легендарного нападающего сборной Голландии, а в компании принято представляться по имени: «Саша-программист», «Паша-программист».

Но каждый раз, когда Йохан звонит новому пользователю и представляется, я представляю, что в этот момент происходит на другом конце провода…

* * *

Сегодня молодой человек, одетый в малиновый пиджак, горячо благодарил меня за то, что полтора года назад я уволил его с должности оператора.

* * *

Лучшим сообщением программы времен MS DOS было:
Invalid user. Replace and strike any key.
А сейчас, конечно же:
User was successfully ignored.

* * *

Кандидат пишет в резюме: «Не имею вредных привычек».
Он не догадывается, что в нашей фирме вредной считается привычка ходить домой ночевать.

* * *

И по прочтении фразы «Ваши предложения прошу высылать в почту» бросил в мусорную корзину очередное резюме.

* * *

– Я сейчас еще подумаю, и скажу вам, какую программу вы должны были написать вчера.

* * *

Ощущения, которые я испытываю, когда понимаю, что забыл дома мобильник (на 80 % оплачиваемый фирмой), больше всего похожи на ощущения при потере невинности: смущение, перемешанное с неимоверным облегчением.

* * *

– Мы сегодня собрались, поскольку у сотрудников накопилось много вопросов, и по офису начали циркулировать различные слухи. Мы ответим сегодня на все ваши вопросы. Мы

не уйдем, пока не ответим на каждый ваш вопрос.

Но ответ на все вопросы у нас один: мы пока не знаем.

* * *

А знаете самый простой способ синхронизации любых баз данных? Это полное их уничтожение.

Приложение 3. Страшный сон в приказах

ПРИКАЗ

№____ «__» _____20_ г.

В связи с вводом в эксплуатацию новой информационной системы предприятия

ПРИКАЗЫВАЮ с «__» _____20_ г.:

1. Заморозить структуру предприятия.

1.1. Запретить переименование подразделений и смену уровня подразделений (превращение отделений в отделы, отделов в управления, управлений в департаменты и обратно).

1.2. Допускается только ликвидация подразделений с увольнением всех сотрудников и создание новых подразделений с набором нового штата.

1.3. Запретить передачу дел при расформировании подразделений.

2. Запретить сотрудникам предприятия изменять фамилии, имена и отчества.

2.1. Допускается увольнение сотрудников для смены фамилии и прием их на работу под новыми фамилиями.

2.2. Запретить передачу дел при увольнении сотрудников.

3. Во все заключаемые предприятием договора вставлять обязательства контрагента не изменять наименование, организационно-правовую форму и учетные коды до выполнения сторонами обязательств по договору, а также не передавать обязательства по договору третьим лицам. Если контрагент отказывается от такого пункта договора, договор не заключать.

Генеральный директор _____

ПРИКАЗ

№____ «__» _____20_ г.

В дополнение к предыдущему приказу для успешной эксплуатации новой информационной системы

ПРИКАЗЫВАЮ с «__» _____20_ г.:

1. Заморозить заключение предприятием договоров с контрагентами.

2. Прекратить любую иную деятельность, требующую изменения информации, хранящейся в информационной системе.

Генеральный директор _____

Приложение 4. Когда машины были большими

Институт я заканчивал во время интенсивного использования больших (по размеру) машин, называвшихся ЕС ЭВМ.

Кстати, основной объект приложения моего труда за время этого труда поменял сексуальную принадлежность. В моей молодости это был не он – компьютер, а она – машина, официально – электронно-вычислительная машина, или ЭВМ. Штука эта занимала даже не одно, а обязательно два больших помещения, называвшихся машинным залом. В одном помещении находились шкафы с пультом (щитом с рядами лампочек и тумблеров или кнопок), процессором, оперативной памятью и системой питания, стол оператора с консольной пишущей машинкой, устройства ввода с перфокарт и перфоленты и вывода на перфоленту и перфокарты (почему-то называвшийся бармалеем), печатающее устройство (то, что теперь называется принтер). Во втором помещении, называвшемся гермозоной, были устройства для работы с магнитными лентами (их звали лентопротяжками) и с магнитными дисками (дисководы). Все шкафы постоянно жужжали и свистели вентиляторами охлаждения, но к этому быстро привыкали. Привыкали настолько, что когда я попал в машинный зал во время профилактических работ, то страшно испугался, но долго не мог понять, чего именно. Потом понял: ужас вызвало то, что в зале было тихо.

В машинный зал полагалось заходить в белых халатах. Остальные требования к одежде мы сформировали сами экспериментально: одежда не могла быть ни шерстяной, ни синтетической, а на ноги надевались тапочки на кожаной подошве. Нарушение этих правил каралось ощутимыми ударами статических зарядов с зависанием машины.

По полу машинного зала проходило огромное количество кабелей, соединявших все устройства, поверх которых был настелен следующий уровень пола, называвшийся фальшполом.

Большинство прикладных программ тогда отлаживалось в пакетном режиме: в окошко сдавали пачки перфокарт с текстом программы и исходными данными для ее работы и через определенное время (один-два раза в сутки) получали результаты трансляции или работы программы, распечатанные на бумаге. Окошко традиционно называлось дуплом, пачка перфокарт – колодой, а напечатанный протокол работы программы – распечаткой.

Избранные программисты, системщики (те, кто занимались поддержанием и развитием операционной системы) и разработчики АСУ, допускались до работы непосредственно в машинном зале. Нам для этого выделялось исключительно ночное время: днем операторы прогоняли пакет. Работать по ночам было вполне романтично, но утомительно.

Программы писались на бумаге карандашом (чтобы можно было стирать ошибки), потом переписывались на бланки кодирования и сдавались в дупло для перфорации (нанесения на перфокарты). Так же, через бланки и дупло, на перфокарты переносились и исходные данные.

Поскольку при таком режиме на замене даже одной перфокарты терялся рабочий день, те ошибки, которые было возможно, исправлялись прорезыванием на перфокарте дополнительных дырок с помощью лезвия безопасной бритвы «Нева» или «Балтика». Говорят, что в некоторых ВЦ умельцы также заклеивали на перфокартах лишние дырки, но у нас это не практиковалось.

Твердого знака и буквы «ё» на клавиатуре не было ни у перфоратора, ни у консольной пишущей машинки. Раскладка русских букв была ЙЦУКЕН и почти совпадала с раскладкой портативной пишущей машинки, а вот латинские буквы располагались под аналогичными русскими (A-A, У-U, Ф-F, H-N Х-H, Ц-C,Й-J, Ы-Y), а буквы, не имевшие аналогов, были раскиданы произвольно. Так, буква Q была на одной клавише с Я, а что было вместе с Ж и Щ, я уже не помню.

Дисплеи массово появились уже позже, через несколько лет, символы на них были зелеными на черном фоне, а строки плавали одна относительно другой, так что через два часа работы перед глазами начинал плыть весь мир.

Принтер назывался АЦПУ (алфавитно-цифровое печатающее устройство), размещался в тумбе внушительных размеров, печатал в одной строчке за один удар 120 — 128 символов на широкой ленте с дырочками по бокам, которая называлась перфорированной бумагой. Бумага такая была в дефиците, выписывалась по разнарядкам, и у нас ее вечно не хватало. Наконец кулибины нашего ВЦ модифицировали на АЦПУ механизм протяжки бумаги, и мы стали печатать на рулонах оберточной бумаги жуткого качества. Зато в соседнем с институтом гастрономе в перфорированную бумагу заворачивали селедку.

АЦПУ в открытом состоянии при работе грохотало настолько сильно, что при диспансерном медицинском осмотре сотрудников нашего ВЦ почти у всех было зафиксировано снижение слуха. Но

не слишком сильное.

Буквы в этом мире не делились на прописные и строчные, они были все одного размера и начертания, тем более не было никаких средств ни для ввода, ни для вывода графической информации. Но это не мешало умельцам рисовать картины из символов на АЦПУ. Впрочем, картин таких было не слишком много, и сейчас я могу вспомнить только алфавитно-цифрового Христа, который висел почти во всех вычислительных центрах.

Аппаратов типа «Ксерокс» было еще очень мало, и охранялись они не хуже огнестрельного оружия. Для тиражирования любой документации (как чертежей, так и текстов) ее сначала наносили на кальку и затем светокопировали. Светокопии получались синими, сизыми, коричневыми, розовыми, фиолетовыми и потому назывались «синьками».

Теперь, вооружив читателя необходимой для восприятия терминологией, можно хоть что-то рассказать.

К свету

Вечер. Плохо освещенный коридор института, только открытое окошко для общения с операторами светится ярко. У него стоит мой знакомый, который работает на дружественную лабораторию. Я здороваюсь и обнаруживаю, что парень близок к истерике.

– Что случилось?

– Не могу найти ошибку, – отвечает он, а губы его трясутся. – Сообщение об ошибке есть, а ошибку найти не могу. Помоги, а?

– Конечно, помогу. А все остальное нормально? Все живы? Никто не заболел?

– Все остальное нормально. Но ты мне поможешь?

– Показывай. Только темновато тут. Давай подойдем к свету.

Мы подходим под лампу, он разворачивает распечатку и неожиданно произносит:

– Спасибо тебе огромное, ты мне так помог.

– Да я еще не помог, ты хоть программу покажи.

– Это уже не надо, я все нашел. Спасибо, – и бежит по направлению к своей лаборатории.

Всего-то и нужно было – подойти к свету…

Поиск ошибок издали

Кажется, я еще студентом был, но программистом уже стал.

То есть это курс пятый, скорее всего. А очкариком я был с детства, и к тому моменту было у меня по минус четыре диоптрии не считая астигматизма.

Сижу с шефом за его столом, обсуждаю теоретические основы реляционной алгебры: «Отношение – это подмножество декартова произведения… Связями называются неупорядоченные отношения», – занятно. Вдруг в дверь, находящуюся на противоположном конце комнаты, врывается возбужденный юноша, разворачивает распечатку во весь свой рост, потрясает ею и восклицает:

– Ну кто, кто сможет найти здесь ошибку?

Я поднимаю голову, тычу пальцем в распечатку и небрежно говорю:

– У тебя там апострофа не хватает. Если поближе подойдешь, покажу место точно, а так далековато, я вижу плохо.

– Где не хватает? – юноша тычет в программу ручкой, но к нам не подходит.

– Ниже, ниже. Вот тут.

– Я проверю.

Он садится за ближайший свободный стол, внимательно смотрит в распечатку, потом растерянно подтверждает:

– Не хватает.

Он был так обескуражен, что даже не попросил объяснить фокус. А фокус был совсем прост: транслятор увидел один апостроф, понял, что дальше следует текстовая константа, и попытался понять, где она заканчивается, но так и не смог этого сделать, просмотрев программу до конца. Как следствие он ни одного оператора дальше не определил, и колонка с номерами операторов, которую я видел как черную вертикальную полосу на левой границе распечатки, пропала как раз на строке, в которой стоял непарный апостроф.

Молодым программистам

Все может быть.
Представим, вы умрете,
Вас выгонят,
Сгорите на работе,
Или на базе вас придавит
Свеклой в таре…
Пожалуйста, пишите комментарий.

И помните: будет хуже

Я не помню, как появилась трещина на стекле в комнате нашего отдела. Не то сама, не то кто-то в кого-то что-то кинул. Но она появилась. Мы заклеили трещину скотчем и написали заявку в хозяйственный отдел о замене стекла.

Через полгода, когда в комнату зашел пожилой стекольщик, никто уже не мог вспомнить, когда и кто его вызывал, но трещина была, и мы ее стекольщику показали. Он внимательно рассмотрел стекло и произнес замечательную фразу:

– Я, конечно, могу заменить, но будет хуже.

Десять юных программистов с хохотом попадали на свои столы (благо клавиатур на них еще не было: они появятся на столах программистов только лет через десять), а обиженный стекольщик пояснил:

– Ну ведь действительно будет хуже: вы сейчас заклеили стекло, и из него не дует, потому что оно стоит на замазке. А у меня замазки нет, я вам новое стекло поставлю на гвозди, и будет дуть.

Тридцать лет прошло. Но каждый раз, когда сисадмин озадаченно интересуется, почему я не даю команду установить новую версию операционной системы или почтового клиента на всех компьютерах офиса, я вспоминаю стекольщика.

Определение полноты функционала системы по весу магнитной ленты

Мне впервые поручили самостоятельно разработать небольшую подсистему учета железнодорожных специалистов с высшим и средним специальным образованием, чтобы прикрыть собственную задницу. Приказ о ее разработке исходил из министерства и обсуждению не подлежал, начать эксплуатацию требовали через два месяца, а шансов на это при тогдашних темпах отладки программ практически не было. Работа как раз для молодого специалиста, которому по служебной лестнице скатиться некуда.

Шеф посоветовал использовать разработку своего знакомого, которому для защиты диссертации нужна была справка о внедрении, поэтому он был готов отдать свою систему даром и обеспечить ее сопровождение. В реляционной СУБД был заложен функционал простого описания таблиц и работы с ними, так что, если она случайно работала, нам для реализации проекта оставалось только «подработать напильником по месту»,

организовав входной контроль информации, ее ввод и печать необходимых отчетов.

С мужиком этим встречались мы почему-то в метро. Наверно, кто-то из нас куда-то спешил. Мобильников тогда не было, но нам тем не менее удавалось договариваться о встрече и встречаться со всеми, кто был нужен.

Найти этого человека среди стоявших на платформе было нетрудно: он держал в руках полиэтиленовый пакет с магнитной лентой. Мы поздоровались, он произнес несколько слов, соответствующих моменту, предложил звонить, если что, и отдал мне бобину и переплетенную документацию. То, что документация была на синьке, говорило об определенной тиражности программного продукта.

Ленту он отдал вместе с пакетом, что было большой щедростью (полиэтиленовых пакетов с ручками в те времена в продаже не было, их добывали на иностранных выставках).

Я взвесил пакет в руке, открыл документацию, просмотрел оглавление, потом отыскал одну страницу, глянул на нее и ласково сказал:

– У вас тут описаны теоретико-множественные операции. Но ведь вы их не реализовали, да?

Выражение его лица доставило мне большое удовольствие.

– Да, но как вы…

– Это просто. Когда бы вы дошли до операции *объединения*, вы бы обнаружили, что на клавиатуре нет твердого знака, и на что-нибудь его заменили. А пока он у вас присутствует даже в описании синтаксиса.[5]

Об авторе

Андрей Орлов имел возможность наблюдать развитие информационных технологий, находясь, как пишут в газетах, в гуще событий. В его активе – 30-летний опыт разнообразной работы в этой сфере, от программирования до руководства ИТ-коллективами и ИТ-проектами, в том числе на уровне CIO. Одним из первых проектов была система послерейсового анализа движения локомотивов. Одним из последних – руководство группой отделов, обеспечивающих использование ИТ в центральном офисе, а также многочисленных складах и магазинах по всей России группы компаний «Ж» (розничная торговля обувью и авторской одеждой).

«С Орловым просто и удобно работать, если вы готовы к тому, что вопросы автоматизации станут почти основными вопросами фирмы. Если вы ищете CIO, наверное, так и должно быть. Приятно то, что уже несколько лет после его ухода система работает и „даже выдает верные результаты“» – говорит один из его работодателей.

За то время, пока информационные системы эволюционировали от «игрушки» для директора завода до жизненно необходимого элемента бизнеса, Андрей Орлов получил второй разряд по альпинизму и первый – по водному туризму. И написал книгу о том, какие «грабли» ожидают «участников увлекательного процесса автоматизации предприятий – от генерального директора до оператора по подготовке данных», и как на них не наступать.

notes

Примечания

1

Ныне именуемые Microsoft Dynamics. – *Д. К.*

2

А годом ранее Navision приобрела компанию Damgaard, которая и была разработчиком Axapta. Такие вот дела. – *Д. К.*

3

Мне встречался специалист, рекомендовавший почаще форматировать винчестер. – *Д. К.*

4

Наверное, желание во что бы то ни стало работать на софте и компьютерах самой последней модели – порождение маркетинговых усилий поставщиков программ. Кто мне может сказать, что полезного добавлено в Microsoft Office за последние десять лет? – *Д. К.*

5

На клавиатурах устройств для ЕС ЭВМ не было твердого знака. Поэтому в машину нельзя было ввести слово «объединение». Если бы функция была реализована, разработчикам пришлось бы как-то решить эту задачу: или писать слово с мягким знаком вместо твердого, или использовать двойную кавычку, или заменить само ключевое слово. Но тогда бы они и в документации привели то название операции, которое на самом деле вводили.